# COLLEEN

# HARDWARE

# MANUAL

## INTRODUCTION

Colleen is the code name for a video game-home computer product that contains a 6502 microprocessor, 4 I/O chips, operating system ROM, and expandable RAM, and several MSI chips for address decoding and data bus buffering.

This manual is intended to primarily describe the 4 I/O chips in sufficent detail to allow experienced programmers to create the operating system code and to create assembly language application ROMS, such as video games. All 4 Input-Output chips are controlled by the microprocessor by writing directly into their registers which are decoded to exist in microprocessor memory space just as RAM does. These I/O chips can also be interogated by the microprocessor by reading simmilar registers.

It is really not necessary for the programmer to know which I/O functions are performed by which of the 4 chips, however it does help in learning these functions.

| CHIP NAME | FUNCTION |
|---|---|
| ANTIC | DMA(direct memory access) control.<br>NMI(non maskable interrupt) control.<br>Vertical and Horizontal fine scrolling<br>Light pen position registers<br>Vertical line counter<br>WSYN(wait for horiz. sync) |
| CTIA | Priority control (display of overlaping objects)<br>Color- Lum control(colors and brightness assigined to all objects including DMA objects from ANTIC)<br>PLAYER-MISSILE objects (4 players & 4 missiles)<br>Graphics registers<br>Size control<br>Horiz. position control<br>Collision detection between all objects<br>Switches and triggers(misc. I/O functions) |
| POKEY | Keyboard scan and control<br>Serial communications port (bidirectional)<br>Pot scan ( digitizes position of 8 independent pots)<br>Audio generation(4 channels)<br>Timers<br>IRQ( maskable interrupt) control<br>Random number generator |
| PIA | Controller(Joy stick) jacks read or write<br>Peripheral control and interrupt lines<br>IRQ( maskable)interupt control from peripherals |

The next few pages will introduce some of the concepts needed to understand the Colleen I/O system.

DMA (Direct Menory Access)

The primary function of the Antic chip is to fetch data from memory (independent of the microprocessor) for display on the TV screen. It does this with a technique called"direct memory access" or DMA. It requests the use of the memory address and data bus by sending a signal called HALT to the microprocessor, causing the processor to become "TRI-STATE" (open circuit) all during the next computer cycle. The ANTIC chip then takes over the address bus and reads any data it wishes from memory. Another name for this type of DMA is "cycle stealing.

Once initiated, this DMA is completely and automatically controlled by the Antic chip without need for further microprocessor intervention. The DMA control circuit on the Antic chip resembles a small dumb microprocessor. By halting the main microprocessor it can fetch it's own instructions from memory(the display list) addressed by its program counter (display list pointer). Each instruction defines the type (alpha character or memory map),and the resolution(size of bits on the screen),and the location of data in memory,to be displayed on the next group of lines.

In order to begin this DMA the main microprocessor must store a display list of instructions in memory, store data to be displayed in memory, tell the Antic where the display list is(initalize the display list pointer) and enable the DMA control flags on the Antic(DMACTL register).

In addition to the type of DMA described above,that is used to generate Alpha Numeric characters and memory map (playfield) displays, the Antic chip simultaniously controls another DMA channel. This type of DMA addresses PLAYER-MISSILE graphics data stored in memory and passes the graphics data on to the CTIA chip graphics registers. This type of DMA (if enabled) occurs automatically interspersed with the playfield DMA described previously. This PLAYER-MISSILE DMA has no display list or instructions, and is therefore much simplier than the PLAYFIELD DMA.

In order to begin PLAYER-MISSILE DMA the main microprocessor simply tells the Antic chip where the data is located in memory (loads the player-missile base register PMBASE) and enables the proper DMA control flags on the Antic chip (DMACTL reg.) and on the CTIA chip (GRACTL reg.).

In addition to the two types of DMA described above,the Antic chip also generates DMA addresses for the refresh of the dynamic memory RAMS used in this system. This is also completly automatic and need be considered by the programmer only if he is concerned with real time programming where an exact count of the computer cycles remaining(after the 3 types of DMA have taken their cycles) is important.

The data fetched by the Antic with PLAYFIELD DMA (Alpha characters or memory map) is stored in a shift register and converted into real time serial output for transmission to the CTIA chip where it is assigined a color-luminance,and compared against other displayed objects for collision detection and priority assignment.

The data addressed by Antic PLAYER-MISSILE DMA(players and missiles) is routed directly to shift registers on the CTIA chip where it is converted into real time serial data representing individual players and missiles,which are assigined color-lum values and compared against each other and Playfield for collision detection and priority assignment.

OBJECTS

There are basically two types of objects produced by the I/O chips for display on the TV screen ; graphics objects and playfield objects. Area on the screen where there are no objects is called Background. Background is not the same as blank or black. It is simply the area where objects are not.

Graphics objects are further divided into Players and Missiles. They are limited in width to 8 bits for each player and 2 bits for each missile. Their vertical height is unlimited. Their horizontal position on the screen is determined by a horizontal position register for each object. Player-Missile data can be fetched from memory by the microprocessor or by the Antic chip (using Player-Missile DMA). This data is then stored by the microprocessor(or automatically by DMA) in registers on the CTIA chip where it is held and outputted to the TV screen whenever the horizontal sync counter equals one of the horizontal position registers. Players and missiles will appear as vertical bars unless their data registers are changed by the microprocessor(or by the data in memory if in Player Missile DMA) during the actual screen display time.

Playfield objects are further divided into Memory map and Characters. Unlike Player-Missile objects that can be moved by simply changing their horiz.position register,Playfield objects have a location on the screen that is determined by their location in memory, and by parameters stored by the microprocessor in the DMA display list in memory. Memory Map playfield data is fetched from memory automatically by the Antic chip where it is placed in a shift register and converted to serial output for the TV display. Character playfield data,in contrast, requires two fetches from memory by the Antic chip. First the Antic fetches the names of the characters from memory and places them in a shift register. These character names are then used to address the actual data which is fetched from memory and converted to serial output for the TV display. All playfield serial output data passes through the CTIA chip before being sent to the TV display.There it is assigined Color-luminance and Priority and tested for Collisions with Graphics objects

COLOR LUM

A color luminance register is used on the CTIA chip for each Player-Missile and Playfield type. Each Color-lum register is loaded by the microprocessor with a code representing the desired color and luminance of it's corresponding Player-Missile or Playfield type. As the serial data passes through the CTIA chip it is "impressed" with the color and luminance values contained in these registers, before being sent to the TV display.

PRIORITY

When moving objects such as players and missiles , overlap on the TV screen(with each other or with Playfield)a decision must be made as to which object shows in front of the other. Objects which appear to pass in front of others are said to have Priority over the ones they pass in front of. Priority is assigned to all objects by the CTIA chip before the serial data from each object is combined with the other objects and sent to the TV screen.

The priority of objects can be controlled by the microprocessor by writing into the control register PRIOR. The functions of the bits in this register are given in the table on page B4.

COLLISIONS

Overlaping objects are considered to have collided. This is detected by a real time occurance of simultanious serial data from more than one object generator. Hardware register bits are used to store 60 of the possible 72 collisions. These collision bits can be read by the Microprocessor as described on pg.11 and B6.

INTERRUPTS

Interrupts are described extensively on pg. 20. Below is a brief list to itemize the types of interrupts provided.

Instruction interrupt.(requested by any display instruction)
Vertical Blank int. (req. by beginning of vertical blank)
Reset button int. (req. by pushing reset button on panel)
Break key int. (req. by pushing break key)
Other key int. (req. by pushing any key)
Serial input int. (req. by serial port input)
Serial output int. (req. by serial port output)
Transmission finished int. (req. by serial port output)
Timer interrupts (3 each, req. by audio timers)
Peripheral interrupts (2each, req. by serial port devices)

Almost all of these interrupt sources can be masked on command of the microprocessor and have status bits which can be interogated and reset by the microprocessor. Even the interrupts defined as "non maskable" (NMI) on the microprocessor have mask bits on the I/O chips which can be set by the microprocessor.

WSYN

In addition to a Vertical Blank Interupt, which allows the microprocessor to synchronize to the vertical TV display, this system also provides a Wait for Horizontal Sync (WSYN) command that allows the Microprocessor to synchronize itself to the TV horizontal line rate. This sync takes effect when the processor writes to an I/O location called WSYN, whenever it desires horizontal synchronization. Writing to this address sets a latch which pulls to zero a pin on the microprocessor called READY. When READY goes to zero the microprocessor stops and waits. The latch is automatically reset(returning READY true) at the beginning of the next horizontal blank interval,releasing the microprocessor to resume program execution.

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H SCROLL | | X | | X | | X | | X | | X | | X | | X | | X | Horizontal Scrolling |
| V SCROLL | | | X | X | | | X | X | | | X | X | | | X | X | Vertical Scrolling |
| LD MEM SCAN | | | | | X | X | X | X | | | | | X | X | X | X | Load memory scan (3 byte) |
| INST INTERUPT | | | | | | | | | X | X | X | X | X | X | X | X | Display instruction interupt |
| BLK 1 | 00 | | | | | | | | 80 | | | | | | | | Blank 1 line |
| " 2 | 10 | | | | | | | | 90 | | | | | | | | Blank 2 lines |
| " 3-7 | [illegible] | | | | | | | | [illegible]0 | | | | | | | | Blank 3 thru 7 lines |
| " 8 | 70 | | | | | | | | F0 | | | | | | | | Blank 8 lines |
| JMP | 01 | | | | | | | | 81 | | | | | | | | Jump (3 byte instruction) |
| JVB | 41 | | | | | | | | C1 | | | | | | | | Jump & wait for vert.Blank (Also 3 Byte) |
| CHR(40,2,8) | 02 | 12 | 22 | 32 | 42 | 52 | 62 | 72 | 82 | 92 | A2 | B2 | C2 | D2 | E2 | F2 | Character Mode Instructions |
| " (40,2,10) | 03 | 13 | 23 | 33 | 43 | 53 | 63 | 73 | 83 | 93 | A3 | B3 | C3 | D3 | E3 | F3 | |
| " (40,4,8) | 04 | 14 | 24 | 34 | 44 | 54 | 64 | 74 | 84 | 94 | A4 | B4 | C4 | D4 | E4 | F4 | |
| " (40,4,16) | 05 | 15 | 25 | 35 | 45 | 55 | 65 | 75 | 85 | 95 | A5 | B5 | C5 | D5 | E5 | F5 | |
| " (20,5,8) | 06 | 16 | 26 | 36 | 46 | 56 | 66 | 76 | 86 | 96 | A6 | B6 | C6 | D6 | E6 | F6 | |
| " (20,5,16) | 07 | 17 | 27 | 37 | 47 | 57 | 67 | 77 | 87 | 97 | A7 | B7 | C7 | D7 | E7 | F7 | |
| MAP(40,4,8) | 08 | 18 | 28 | 38 | 48 | 58 | 68 | 78 | 88 | 98 | A8 | B8 | C8 | D8 | E8 | F8 | Memory Map Mode Instructions |
| " (80,2,4) | 09 | 19 | 29 | 39 | 49 | 59 | 69 | 79 | 89 | 99 | A9 | B9 | C9 | D9 | E9 | F9 | |
| " (80,4,4) | 0A | 1A | 2A | 3A | 4A | 5A | 6A | 7A | 8A | 9A | AA | BA | CA | DA | EA | FA | |
| " (160,2,2) | 0B | 1B | 2B | 3B | 4B | 5B | 6B | 7B | 8B | 9B | AB | BB | CB | DB | EB | FB | |
| " (160,2,1) | 0C | 1C | 2C | 3C | 4C | 5C | 6C | 7C | 8C | 9C | AC | BC | CC | DC | EC | FC | |
| " (160,4,2) | 0D | 1D | 2D | 3D | 4D | 5D | 6D | 7D | 8D | 9D | AD | BD | CD | DD | ED | FD | |
| " (160,4,1) | 0E | 1E | 2E | 3E | 4E | 5E | 6E | 7E | 8E | 9E | AE | BE | CE | DE | EE | FE | |
| " (320,2,1) | 0F | 1F | 2F | 3F | 4F | 5F | 6F | 7F | 8F | 9F | AF | BF | CF | DF | EF | FF | |

Number of TV lines per cell
Number of Colors(Background +Playfield types)
Number of Horizontal cells (Std.width screen)

D I S P L A Y  I N S T R U C T I O N  O P  C O D E S

## VERTICAL AND HORIZONTAL FINE SCROLLING

Playfield objects are difficult to move smoothly. Memory map playfield can be moved by rewriting sections of memory,however this is extremely time consuming if large sections of the screen must be moved smoothly. Character playfield objects can be moved easily in a jerkey fasion by changing the memory scan counter, however this results in a large position jump from one character position to another,not a smooth motion. For this reason hardware registers and counters are provided to allow smooth horizontal or vertical motion, up to one character width horizontally and up to one character height vertically. After this much smooth motion has been done by increasing the value in these registers,memory is rewritten or the memory scan counter is modified and smooth motion is resumed for another character distance. The details of the use of these registers is given on pg.12 and pg.A4

## LIGHT PEN

A "light pen" input is provided which is connected to the Antic chip. This light pen signal captures the value of the vertical line counter and the horizontal sync counter in two registers, PENV and PENH , whenever the signal goes from true to false because of light falling on the light pen. The microprocessor can then read these two registers to determine the pen vertical and horizontal position.

## VERTICAL LINE COUNT

The microprocessor can also read a location that contains the present TV line number being displayed. This allows the microprocessor to modify the display depending on the present vertical location of the TV spot creating the display.

## OBJECT GENERATION

Objects can be generated either as playfield or as player-missile graphics. (see pg 2) These are two distinct, almost independent, object generating circuits.

## PLAYER-MISSILE GRAPHICS GENERATION

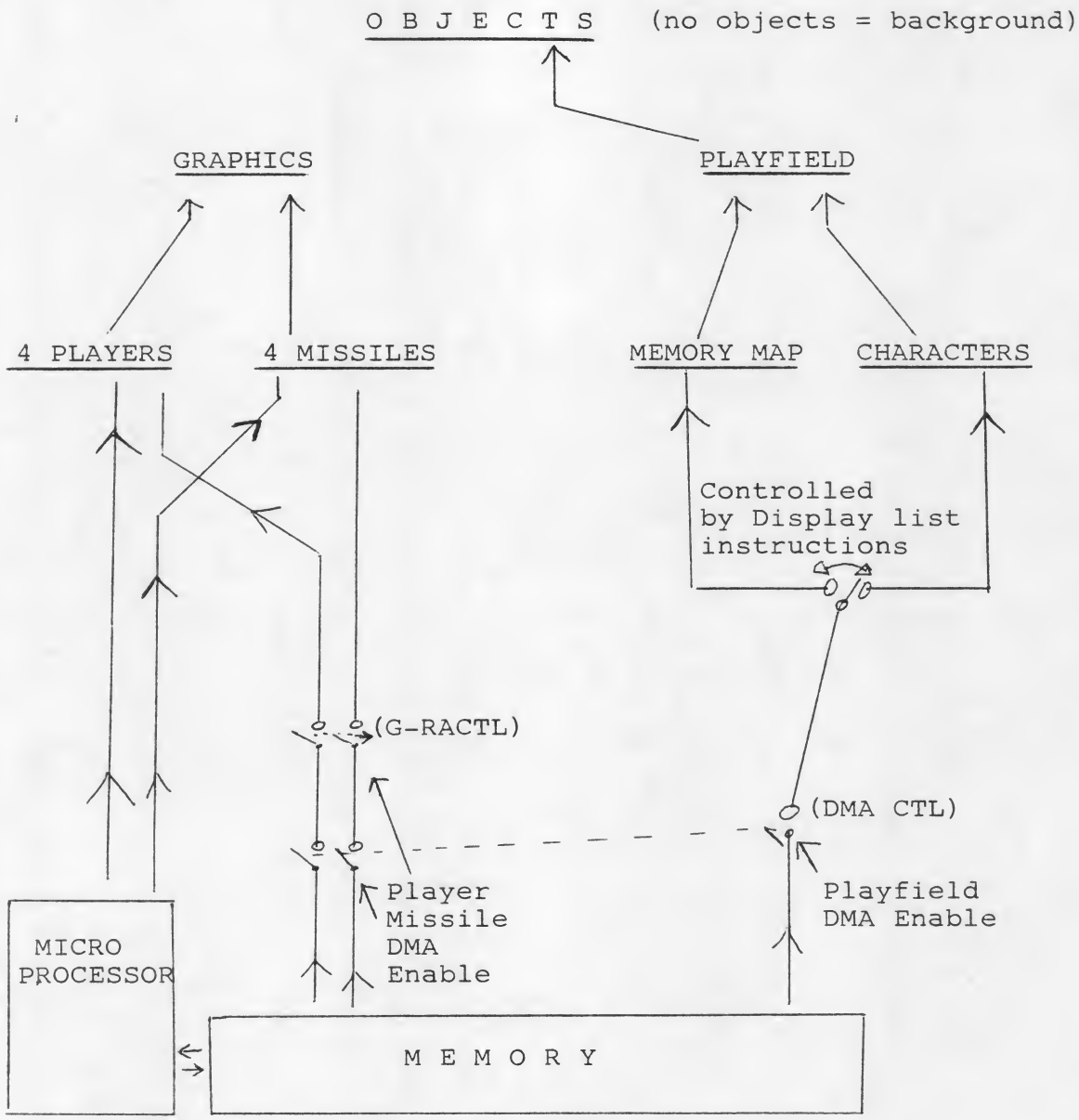
There are 8 graphic objects, 4 players and 4 missiles, the 4 missiles may be grouped together and used as a 5th player, these objects are positioned horizontally by 8 horz. position registers. (HPOS (X)). These registers may be reloaded at any time by the processor, allowing an object to be replicated many times across a horizontal TV line.

The shape of a player-missile, is determined by the data in its graphics register (GRAF(X)). Players have independent 8 bit graphic registers. The four missiles have 2 bit registers (located within one address). These registers may also be reloaded at any time by the processor, although they are usually changed during horizontal blank time. The data in each graphics register is placed on the display whenever the horizontal sync counter equals the corresponding horizontal position register. The same data will be displayed every line unless the graphic registers are reloaded with new data.

The player-missile graphic registers may be reloaded by the microprocessor (GRAF(X)), or automatically directly from memory with direct memory access (DMA). The programmer must place the object graphics in memory, (see pg 3). write the player-missile base address (PMBASE), and enable player-missile DMA (DMACTL, GRACTL). The transfer of object graphics from memory to display is then fully automatic.

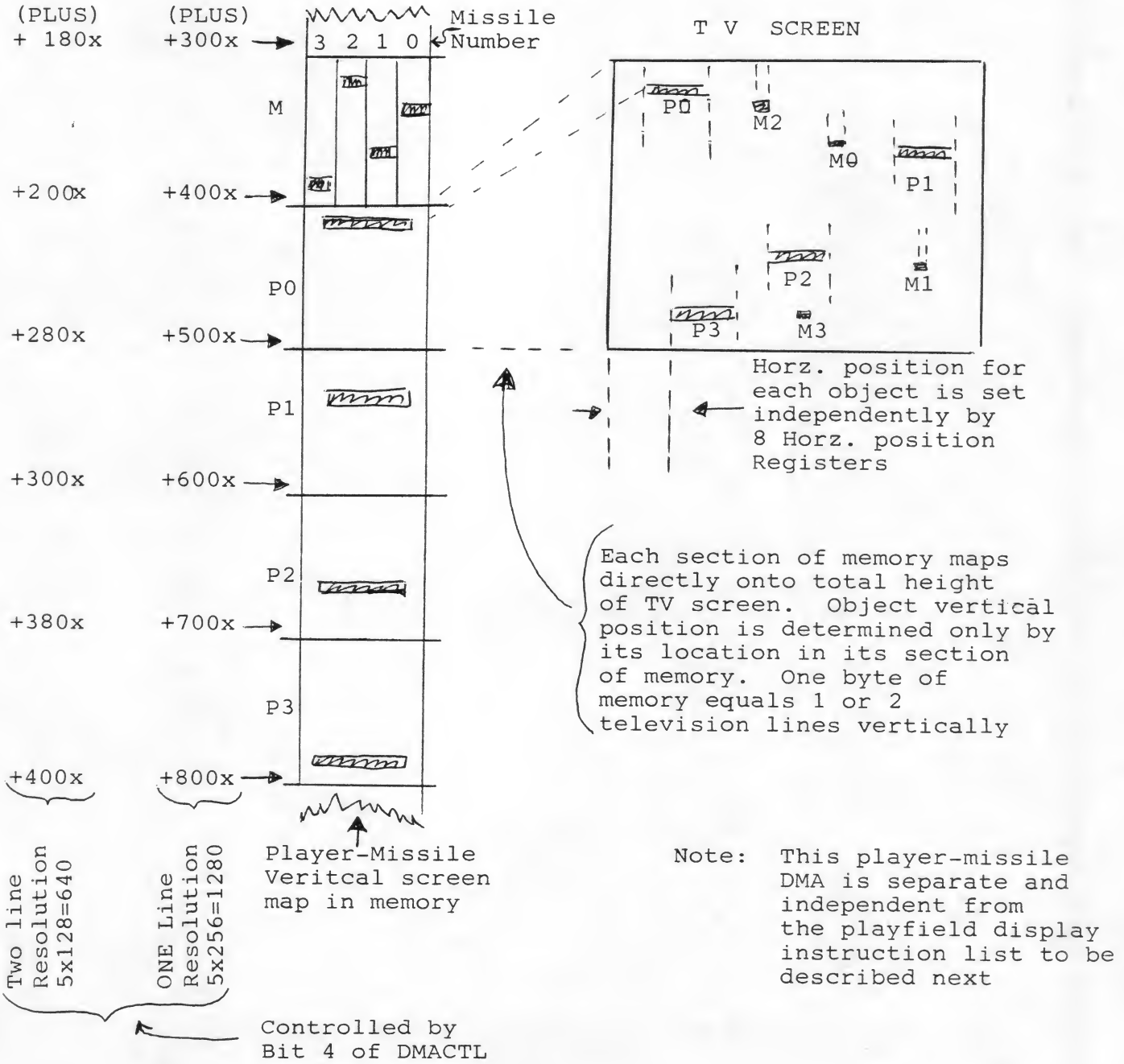
## PLAYFIELD GENERATION

Playfield is always generated by DMA. There are 4 types of playfield, each identified by its own color-lum register and collision detection. Playfield is generated by two different DMA techniques; memory map and character. Both methods provide list of instructions in memory, independent of the player-missile generation.




O B J E C T D I S P L A Y S O U R C E S

Player-Missile
Base Address (PMBASE)
Nx1024 Nx2048




Note: This player-missile DMA is separate and independent from the playfield display instruction list to be described next

P L A Y E R - M I S S I L E D M A

## PLAYFIELD DMA

### Display List

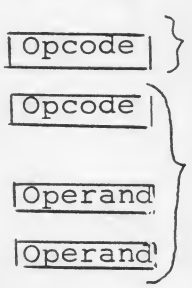
The display list is a sequence of display instructions stored in memory. These instructions are either one Byte, or 3 Bytes long. The display list can be considered a display program, and the Display List Counter that fetches these instructions can be thought of as a display program counter. (10 bit counter plus 6 bit base reg.)

The display list counter can be initalized at any time by writing to DLISTH and DLISTL . Once initalized this counter value is used to address the display list, fetch the instruction, display 1 to 16 lines of data on the TV screen, increment the display list counter, fetch the next display instruction, and so on automatically without microprocessor control. (see pg A1 for dlist bits)

Each instruction defines the type (alpha character or memory map) and the resolution (size of bits on screen) and the location of data in memory to be displayed, for a group (1 to 16) of lines. Each group of lines is called a display block.

### DISPLAY INSTRUCTION FORMAT

Each intruction consists of either an Opcode only, or of an Opcode followed by 2 Bytes of operand.




Sincle Byte Display Instruction

Triple Byte Display Instruction

The Opcode is always fetched first and placed in the Instruction Register. This Opcode defines the type of instruction (one or three Byte) and will cause two more bytes to be fetched if needed. If fetched, these next 2 bytes will be placed in the Memory Scan Counter, or in the Display List Counter (if instruction is a Jump).

## DISPLAY INSTRUCTION REGISTER

This register is not directly accessable by the programmer. It is loaded with the Opcode of each instruction.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
|---|---|---|---|---|---|---|---|---|
| X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Blank 1 line |
| X | 0 | 0 | 1 | 0 | 0 | 0 | 0 | Blank 2 lines etc |
| X | 1 | 1 | 1 | 0 | 0 | 0 | 0 | Blank 8 lines |
| 0 | x | x | x | x | x | x | x | No Interrupt |
| 1 | x | x | x | x | x | x | x | Interrupt (bit 7 of NMI status) |
| x | 0 | x | x | 0 | 0 | 0 | 1 | Jump |
| x | 1 | x | x | 0 | 0 | 0 | 1 | Jump and wait (no display) until end of next vertical blank time. (Jumps are 3 bytes and they reload display list counter) |

Actually Background color-lum not black.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | x | x | x | x | x | x | x | No Interrupt |
| 1 | x | x | x | x | x | x | | Interrupt(bit 7 of NMI Statue) |
| 0 | x | x | x | x | x | x | x | One Byte Inst. |
| 1 | x | x | x | x | x | x | x | 3 Byte inst. (Reload Mem. Scan Counter) |
| 0 | x | x | x | x | x | x | x | No Vertical Scroll |
| 1 | x | x | x | x | x | x | x | Vertical Scroll |
| 0 | x | x | x | x | x | x | x | No Horizontal Scroll |
| 1 | x | x | x | x | x | x | x | Horizontal Scroll |

see VSCROL

see HSCROL

Display Mode OPcodes (Jump & Blank Excluded) See list of Display Modes on pg. 7 10 & 11

## MEMORY SCAN COUNTER

This counter is not directly accessable by the programmer. It is loaded with the value in the last 2 Bytes of a 3 Byte (non Jump) instruction.

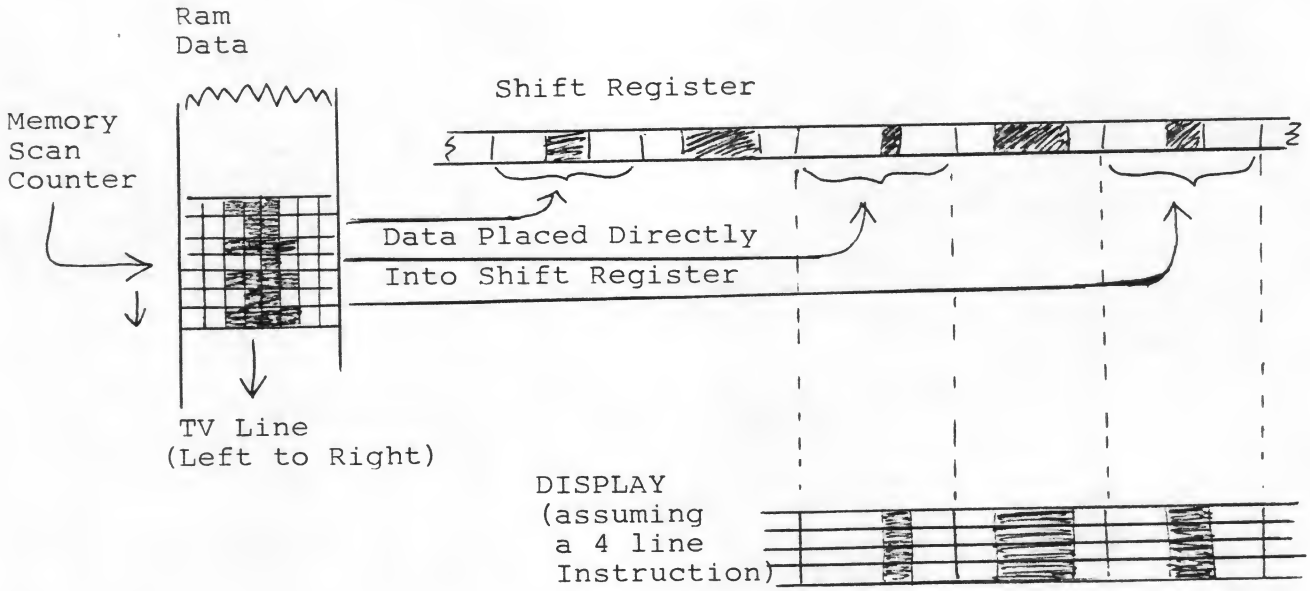
This counter points to the location (address) in memory of data to be directly displayed (memory map display) or to the location of character name strings to be indirectly displayed (character display).

A single Byte instruction does not reload this counter. This implies a continuation in memory of data to be displayed from that displayed by the previous instruction. Since this counter really

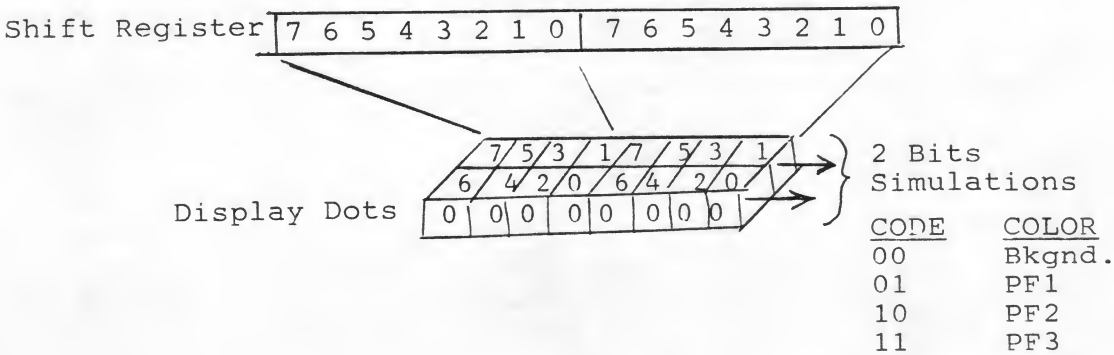
Consists of 4 bits of register and 12 bits of actual counter, a continuous memory block cannot cross 4K Byte memory boundaries, unless the counter is repositioned with a 3 byte type of instruction. (Non Jump 3 byte instruction)

MEMORY MAP DISPLAYS

Display data is fetched directly by the memory scan counter and placed in a shift register. This shift register is used to display as many lines as required by the display instruction.

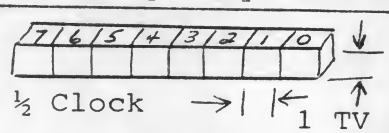
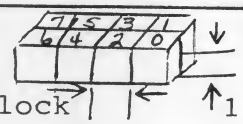
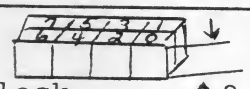
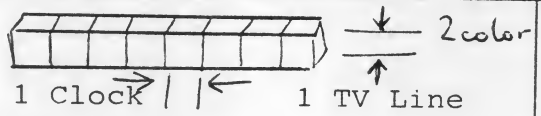
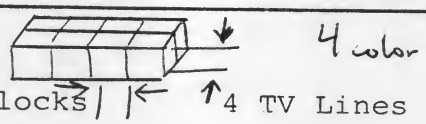
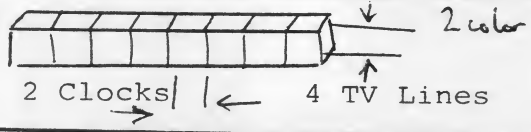
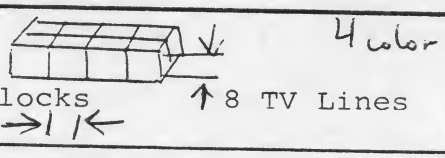



Some instructions compress data in the shift register to give a two bit deep display. The two bits of depth allow each display dat to be identified as background (Both Bits Zero) or as one of 3 types of playfield.



## MEMORY MAP DISPLAY INSTRUCTIONS

Data in memory (addressed by the memory scan counter) is displayed directly, when executing a memory map display instruction. As data is being displayed it is also stored in a shift register so that it can be redisplayed for a many TV lines as required by the instruction.

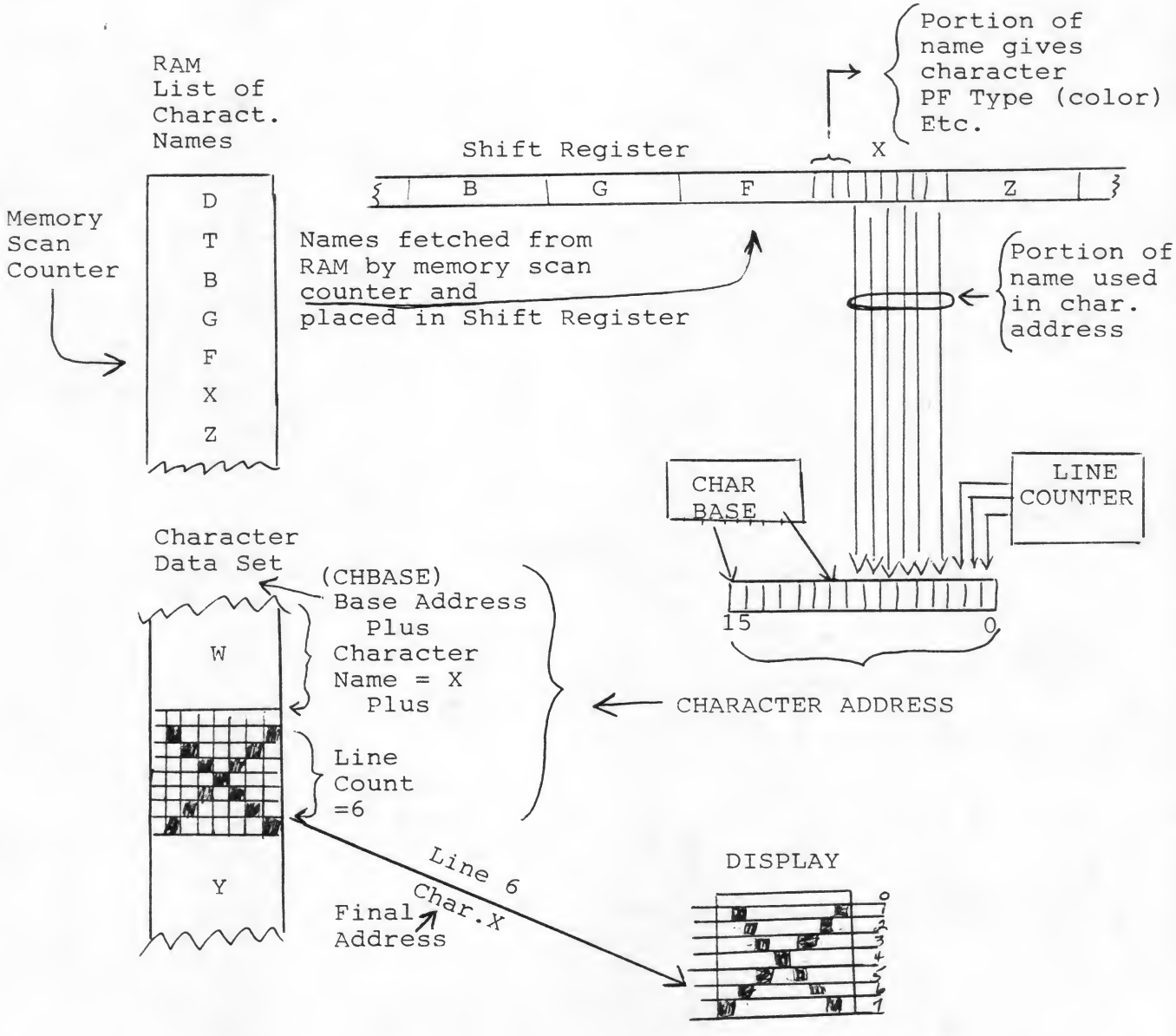
| Basic Graphics Mode | (Instruction Register Bits) 3 2 1 0 | Output Displayed by each Byte | Std. Horiz Bits Displ | Color Lum | Inst. Vert TV Lines |
|---|---|---|---|---|---|
| 8 | 1 1 1 1 |   | 320 | PF2 Ø<br>PF1 1<br>(Lum only) | 1 |
| — | 1 1 1 0 |   | 160 | BK Ø<br>PF1 1<br>PF2 2<br>PF3 3 | 1 |
| 7 | 1 1 0 1 |   | 160 | BK Ø<br>PF0 1<br>PF1 2<br>PF2 3 | 2 |
| — | 1 1 0 0 |   | 160 | BK<br>PF0 | 1 |
| 6 | 1 0 1 1 |   | 160 | BK Ø<br>PF0 1 | 2 |
| 5 | 1 0 1 0 |   | 80 | BK Ø<br>PF0 1<br>PF1 2<br>PF2 3 | 4 |
| 4 | 1 0 0 1 |   | 80 | BK Ø<br>PF0 1 | 4 |
| 3 | 1 0 0 0 |   | 40 | BK Ø<br>PF0 1<br>PF1 2<br>PF2 3 | 8 |

Data

Note: All memory map and character instructions display data as one of 4 types of playfield or background, each with it's own separate color-lum register.

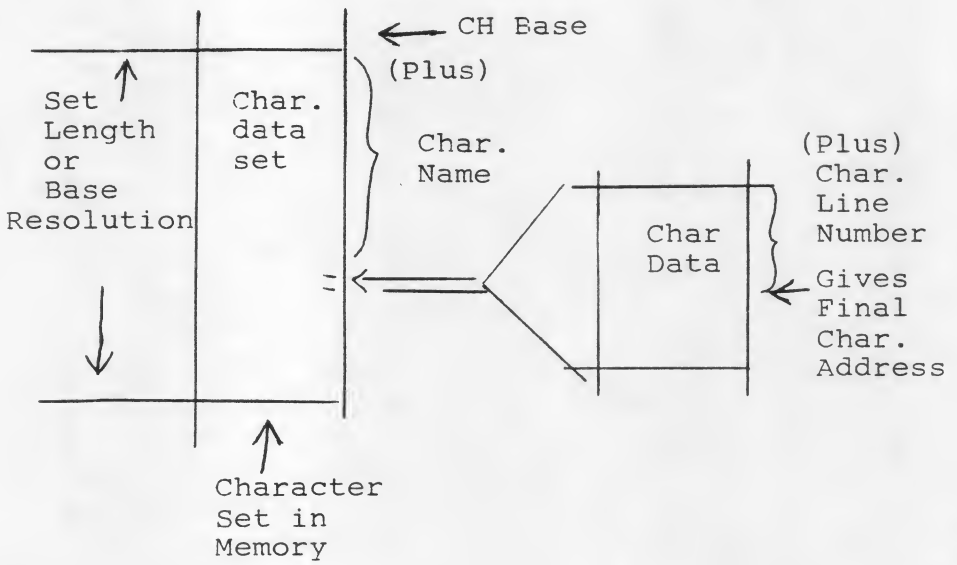
CHARACTER DISPLAYS

Character names (codes) are fetched by the memory scan counter, and are placed in a shift register. On any given line of display, the shift register rotates changing only the name portion of the character address, as shown below.




After a full line of character data has been displayed the △ line counter increments. The next line again addresses all characters by name for that △ line number.

Only the most significant 6 or 7 bits of the character base register are used in the final address, depending on the instruction type. Character sets therefore come in 2 different sizes; 512, and 1024 Total Bytes (Max).




| Char. Instruct. Code | Char. Display Type | Set Length Base Resolution | Number of Char. in set | Bytes Per char. |
|---|---|---|---|---|
| 011X | 20x5 | 512 Bytes | 64 | 8 |
| 010X | 40x4 | 1024 Bytes | 128 | 8 |
| 001X | 40x2 | 1024 Bytes | 128 | 8 |

Characters per line

#of colors

Base = N x Block Length

Different portions of the character name are used in the final address, also depending on the instruction type as shown in the following list of character instructions.

## CHARACTER DISPLAY INSTRUCTIONS

Data in memory (addressed by the memory scan counter) is not displayed directly when executing a character display instruction. The memory scan counter points instead to a list of character names. This list of names might for example contain the ASCII code names for alpha-numeric characters to be displayed. This name list is fetched by the memory scan counter and placed in a shift register. These character names are then combined with the TV line count, and the character base address, to create the character address. That actually fetches character data to be displayed. (see "CHBASE" pg. D40F)

| IR BITS 3 2 1 0 | Output Displayed by each Byte (All Characters are 8 Bytes High) | Std Horz Char Disp | Color lum | Inst. Vert. TV Lines |
|---|---|---|---|---|
| 0 1 1 1<br>(20x5) | 1 Clock<br>Data 7 6 5 4 3 2 1 0 → Background<br>Four PF Codes<br>Name 7 6 5 4 3 2 1 0<br>Used in Character Address | 20 | BK<br>PF0<br>PF1<br>PF2<br>PF3<br>=5 | 16 |
| 0 1 1 0 | Same as above except each data Byte shown for 1 line instead of 2 | 20 | same | 8 |
| 0 1 0 1<br>(40x4) | 1 Clock<br>Data 7 5 3 1 / 6 4 2 0<br>4 codes<br>Name 7 6 5 4 3 2 1 0<br>Alterable PF Selection<br>Used in Char. Add. | 40<br>[11= if or if | BK<br>PF0<br>PF1<br>PF2<br>=4<br>PF3<br>CH 7= PF2<br>bit 7 of | 16<br>1 |
| 0 1 0 0 | Same as above except each data Byte shown for 1 line instead of 2 | 40 | Same | 8 |

data=Ø

Name Bits

name=Ø

## Character Display Instructions

| Instr. Reg. Bits 3 2 1 0 | Output Displayed by Each Byte (Characters are all 8 Bytes High) | Std. Horz Chars Displ | Color Lum | Inst. Vert TV Lines |
|---|---|---|---|---|
| | ½ Clock Data → 1 ← <br> 7 6 5 4 3 2 1 0 → 1 PF Type + Bkgnd. | 40 | PF2 PF1 (Lum only) | 10 |
| 0 0 1 1 <br> (40x2) | Name <br> 7 6 5 4 3 2 1 0 <br> Invert Blank Flag (see CHCTL) <br> Display Character on last 8 Lines if Bit 6.5 (lower case) <br> Used in Character Address | (one fourth of character set) | | |
| 0 0 1 0 | Same as above except only 8 Lines total for each instruction | 40 | Same | 8 |

## HARDWARE COLLISION DETECTION

60 bits of collision register are provided to detect and store overlap (hits) between players, missiles and playfield. These collisions can be read by the microprocessor from addresses D000 through D00F.

16 bits for Missile to Playfield
16 bits for Player to Playfield
16 bits for Missile to Playfield
12 bits for Player to Player (P0 to P0 always reads as zero. ETC.)

The ½ clock memory map mode (IR code 1111) and the ½ clock Character mode (IR codes 0011 and 0010) are both playfield type 2 and collisions will be stored in bit 2 of the playfield collision registers.

## VERTICAL SCROLLING, DETAILS OF OPERATION

For vertical scrolling of a zone of display on the screen, The display blocks at the upper and lower boundries of that zone must have a variable vertical size. In particular, the first display block within that zone must be shortened from the top, and the last display block must be shortened from the bottom.

The vertical dimension of each display block is controlled by a 4 bit counter within the Antic, called the 'delta counter' (DCTR). Without vertical scrolling, it starts at Ø on the first line, and counts up to a standard value, determined by the current display instruction. (Ex: for upper and lower case text display, the end value is 9. For 5 color character displays, it is 7 or 15).

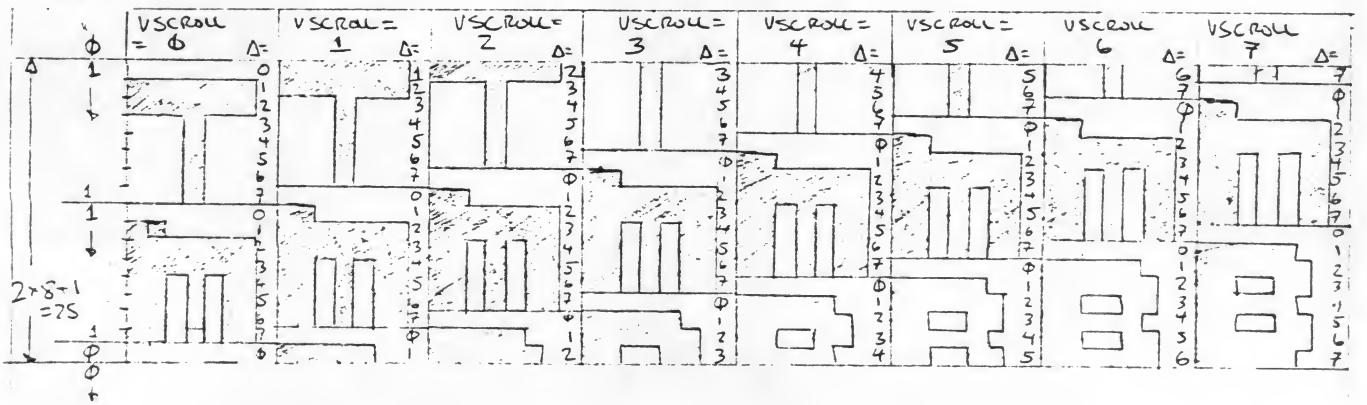
Bit 5 of the instruction controls vertical scrolling
If Bit 5 goes from Ø to 1 between two display blocks, the second block will start with the Delta Counter loaded with the 4 bit value in the vscroll register, instead of Ø, shortening it from the top.

If bit 5 of the instruction goes from 1 to Ø between two consecutive display blocks, the second block will start with Delta = Ø, as usual, but will count up until delta=vscroll, instead of the standard value. This shortens that display block from the bottom.

If bit 5 of the instruction does not change between consecutive display blocks, vertical scrolling does not occur.

To define a vertically scrolled zone, the most direct method is to set bit 5 = 1 in the first display instruction for that zone, and in all consecutive blocks but the last one. If the vscroll register is not rewritten on the fly, this results in a total scrolled zone that has a constant number of lines (provided that the vscroll value does not exceed the standard individual clock size). If N is the standard block size, the top block will be n-vscroll lines (n > vscroll), and the last clock will be vscroll+1 lines: (N-vscroll) + vscroll+1= N + 1. Shown below is an example of a scrolled zone, top block, middle block, and bottom block, for 8 vscroll values for n=8.

IR5



## AUDIO

There are 4 semi-independent audio channels, each with its own frequency, noise, and volume control. Each has an 8 bit "divide by N" frequency divider, controlled by an 8 bit register (AUDFX). (See audio-serial port block diagram). Each channel also has an 8 bit control register (AUDCX) which selects the noise (poly counter) content, and the volume.

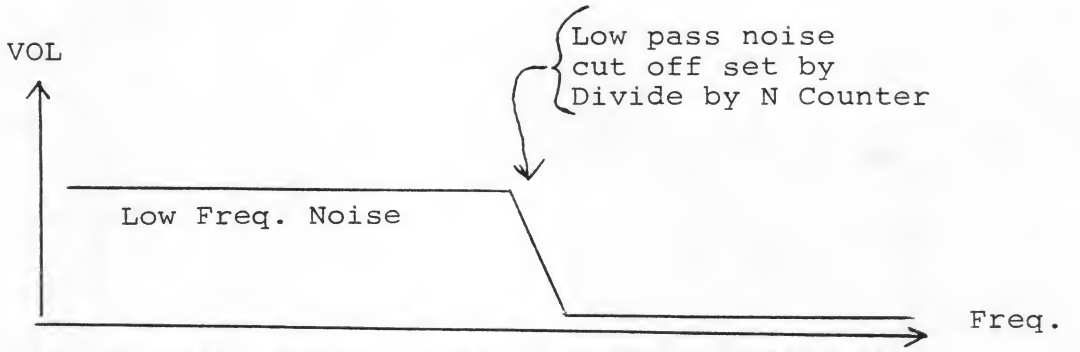
## FREQUENCY DIVIDERS

All 4 freq. dividers can be clocked simultaniously from 64 KHZ or 15 KHZ. (AUDCTL bit 0). Freq. dividers 1 and 3 can alternately be clocked from 1.79 MHZ (AUDCTL bit 6,5). Dividers 2 and 4 can alternately be clocked with the output of dividers 1 and 3 (AUDCTL bits 4,3) This allows the following options; 4 channels of 8 bits resolution, 2 channels of 16 bit resolution, or 1 channel of 16 bit and 2 channels of 8 bit.

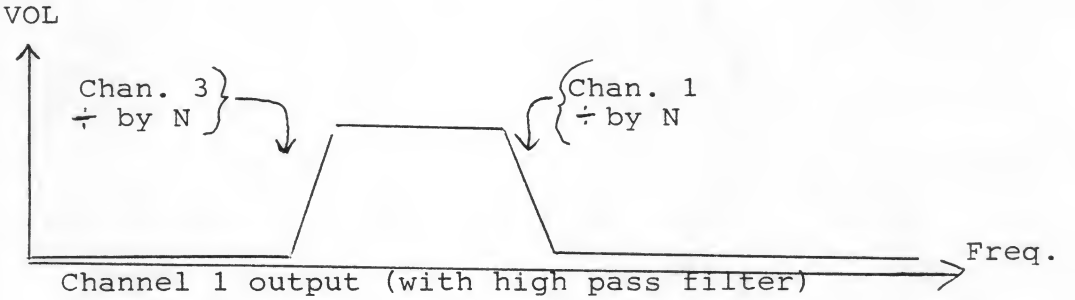
## POLY NOISE COUNTERS

There are 3 polynomial counters (17 bit, 5 bit and 4 bit) used to generate random noise. The 17 bit poly counter can be reduced to 9 bits (AUDCTL bit 7). These counter are all clocked by 1.79 MHZ. Their outputs however can be sampled independently by the four audio channels at a rate determined by each channel's frequency divider. Thus each channel appears to contain separate poly counters (3 types) clocked at its own frequency. This poly counter noise sampling is controlled by bits 5, 6 and 7 of each AUDCX register. Because the poly counters are sampled by the "divide by N" frequency divider, the output obviously cannot change faster than the sampling rate. In these modes (poly noise outputed) the dividers are therefore acting as "low pass" filter clocks, allowing only the low frequency noise to pass.

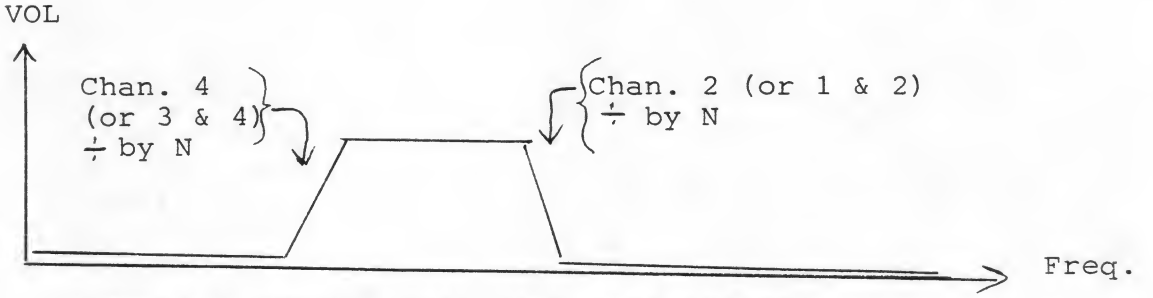
The output of the noise control circuit described above consists of pure tones (square wave type), or polynomial counter noise at a maximum frequency set by the "divide by N" counter (low pass clock). This output can be routed through a high pass filter if desired, (AUDCTL bits 1 and 2).




Any channel noise output (without high pass filter)




Channel 1 output (with high pass filter)




Channel 2 output (with high pass filter)

A U D I O  N O I S E  F I L T E R S

HIGH PASS FILTERS

The high pass filter consists of a "D" flip flop and an exclusive-OR Gate. The noise control circuit output is sampled by this flip flop at a rate set by the "High Pass" clock. The input and output of the Flip Flop Pass through the exclusive-OR Gate. If the flip flop input is changing much faster than the clock rate, the signal will pass easily through the Ex.-OR Gate. However if it is lower than the clock rate, the flip flop output will tend to follow the input and the two Ex. - OR gate inputs will mostly be identical (11 or 00) giving very little output. This gives the effect of a crude high pass filter, passing noise whose minimum frequency is set by the high pass clock rate. Only channels 1 and 2 have such a high pass filter. The high pass clock for channel 1 comes from channel 3 divider. The high pass clock for channel 2 comes from channel 4 divider. This filter is included only if bit 1 or 2 of AUDCTL is true.

VOLUME CONTROL

A volume control circuit is placed at the output of each channel. This is a crude 4 bit digital to analog converter that allows selection of one of 16 possible output current levels for a logic true audio input. A logic zero audio input to this volume circuit always gives an open circuit (zero current) output. The volume selection is controled by bits 0 thru 3 of AUDCX. "Volume Control only" mode can be invoked by forcing this circuit's audio input true with bit 4 of AUDCX. In this mode the dividers, noise counters, and filter circuits are all disconnected from the channel output. Only the volume control bits (o-3 of AUDCX) determine the channel output current.

The audio output of any channel can be completely turned off by writing zero to the volume control bits of AUDCX. All ones gives maximum volume.

## SERIAL PORT

### General

The serial port consists of a serial data output (transmission) line, a serial data input (receiver) line, a serial output clock line, a bi-directional serial data clock line, and other misc. control lines described in the chapter called "Serial Port protocol". Data is transmitted and received as 8 bits of serial data preceeded by a logic zero start bit, and succeeded by a logic true stop bit. Input and output clocks are equal to the baud (bit) rate, not 16 times baud rate. Transmitted data changes when the output clock goes true. Received data is sampled when the input clock goes to zero.

### SERIAL OUTPUT

The transmission sequence begins when the processor writes 8 bits of parallel data into the serial output register (SEROUT)(see audio and serial port block diagram). When any previous data byte transmission is finished the hardware will automatically transfer new data from (SEROUT) to the output shift register, interrupt the processor to indicate an empty (SEROUT)register (ready to be reloaded with the next byte of data), and automatically serially transmit the shift register contents with start-stop bits attached. If the processor responds to the interrupt, and reloads SEROUT before the shift register is completely transmitted, the serial transmission will be smooth and continuous.

Output data is normally transmitted as logic levels (+4V=true OV=False). Data can also be transmitted as two tone information. This mode is selected by bit 3 of SERCTL. In this mode audio channel 1 is transmitted in place of logic true, and audio channel 2 in place of logic zero. Channel 2 must be the lower tone of the tone pair.

The processor can force the data output line to zero (or to audio ch. 2, if in two tone mode) by setting bit 7 of SERCTL. This is required to force a break (10 zeros) code transmission.

SERIAL OUTPUT CLOCK

The serial output data always changes when the serial output clock goes true. The clock then returns to zero in the center of the output data bit time.

The baud (bit) rate of the data and clock is determined by: audio channel 4 audio channel 2, or by the input clock, depending on the serial mode selected by bits 4, 5, 6 of SERCTL. (See chart at end of this section.)

SERIAL INPUT

The receiving sequence begins when the hardware has received a complete 8 bit serial data word plus start and stop bits. This data is automatically transferred to the 8 bit parallel input register (SERIN), and the processor is interrupted to indicate an input data byte ready to read in SERIN. The processor must respond to this interrupt, and read SERIN, before the next input data word reception is complete, otherwise an input data "over-run" will occur. This over run will be indicated by bit 5 of SKSTAT. (If bit 5 of IRQST is not RESET (true) before next input complete), and means input data has been lost. This bit should be tested whenever SERIN is read. Bit 7 of SKSTAT should also be tested to detect frame errors caused by extra (or missing) data bits.

DIRECT SERIAL INPUT

The serial data input line can be read directly by the microprocessor if desired, ignoring the shift register, by reading bit 4 of SKSTAT.

BI-DIRECTIONAL CLOCK

This clock line is used to either receive a clock from an external clock source for clocking transmitted or received data, or is used to supply a clock to external devices indicating the transmit or reception rate. This clock line direction is determined by the serial mode selected by bits 4, 5, 6 of SERCTL. (See mode chart at the end of this section). Transmitted data changes on the rising edge of this clock. Received data is sampled on the trailing edge of this clock.

ASYNCHRONOUS SERIAL INPUT

Unclocked serial data (at an approximately known (± 5%) rate) can be received in the asynchronous modes. The receive (input) shift register is clocked by audio channel 4, channels 3 & 4 should be used together (AUDCTL bit 3=1) for increased resolution. In async. modes, channels 3 and 4 are reset by each start bit at the beginning of each serial data byte. This allows the serial data rate to be slightly different from the rate set by channels 3 & 4.

SERIAL MODE CONTROL

There are 6 useful modes (of the possible 8) controlled by bits 4, 5, 6 of SERCTL. These are described on the next page.

Note that two tone output (bit 3 of SERCTL) may be used in any of these modes except for the bottom pair. This is because chan 2 is used to set the output transmit rate and is therefore not available for one of the 2 tones.

Note that the output clock rate is identical to the output data rate.

SERIAL MODE CONTROL (see also register description ~~SERCTL~~ SKCTL)

Force Break

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

← SKCTL ~~SERCTL~~ REGISTER

Pot scan and keyboard CTRL

Two Tone Control

Mode Control Bits

(A)=asynchronous

| D6 | D5 | D4 | Out Rate | Out Clock | In Rate | Bi-Dir Clock | Comments |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | ext | ext | ext | ext input | Trans. & Receive rates set by external clock also internal clock phase reset to zero |
| 0 | 0 | 1 | ext | ext | chan 4 (A) | ext input | Trans. rate set by external clock. Receive asynch. (ch. 4) (CH3 and CH4) |
| 0 | 1 | 0 | chan 4 | chan 4 | chan 4 | chan 4 output | Trans. & Receive rates set by chan. 4 Chan. 4 output on Bi-Directional Clock Line |
| 0 | 1 | 1 | ch4 (A) | ch4 (A) | ch4 (A) | input | Not Useful |
| 1 | 0 | 0 | chan 4 | chan 4 | ext | ext input | Trans. Rate Set by Chan. 4. Receive Rate set by External Clock |
| 1 | 0 | 1 | ch4 (A) | ch4 (A) | ch4 (A) | input | Not Useful |
| 1 | 1 | 0 | chan 2 | chan 2 | chan 4 | chan 4 Output | Trans. rate set by chan.2 Receive " " " " 4 Chan 4 out on Bi-Direct. Clock line. |
| 1 | 1 | 1 | chan 2 | chan 2 | chan 4 (A) | input not used | Trans. Rate set by chan. 2. Receive async. (chan 3&4) Bi-Dir. Clock not used (Tri-state condition) |

Two tone (bit 3) not useable in these modes

## INTERRUPT SYSTEM

### General

There are two basic types of interrupts defined on the microprocessor; NMI (non maskable interrupt) and IRQ (interrupt request) it is recommended that a thorough understanding of these interrupt types be acquired by reading all chapters concerning interrupts in the 6502 microprocessor programming and hardware manuals.

In this system NMI interrupts are used for video display and reset. IRQ interrupts are used for serial port communication, peripheral device, timers, and keyboard inputs.

### NMI Interrupts

Even though NMI interrupts are "unmaskable" on the microprocessor, this system has interrupt enable ($\overline{\text{mask}}$) bits for NMI function. (Bits 6, 7 of NMIEN) when these bits are zero NMI interrupts are disabled (masked) and prevented from causing a microprocessor NMI interrupt. (see NMIEN register description) The 3 types of NMI interrupts are:

1. D7 = Instruction Interrupt (during display time any display instruction with bit 7=1 will cause this interrupt to occur (if enabled) at the start of the last video line of the mode)

2. D6 = Vertical Blank Interrupt (interrupt occurs (if enabled) at the beginning of the vertical blank time interval.)

3. D5 = Reset Button Interrupt (pushing the front panel reset button will cause this interrupt to occur).

Since any of these interrupts will cause the processor to jump to the same NMI address, the system also has NMI status bits which may be examined by the processor to determine which source caused the NMI interrupt. Bits 5, 6, 7 of NMIST serve this function. (see NMIST register description). These status bits are set by the corresponding interrupt function (even if the interrupt is masked from the processor by NMIEN). The status bits may be reset together by writing to the address NMIRES.

Two of the interrupt enable bits (bits 6, 7of NMIEN) are cleared automatically during system power turn on and therefore these NMI interrupts are initally disabled (masked), preventing any power turn on service routine from being interrupted before proper initalization of registers and pointers. *They can then be enabled by the processor whenever desired, by writing into bits 6 or 7 of NMIEN. Except for the reset button interrupt, they can also be disabled by the processor by writing a zero into bits 6 or 7 of NMIEN. The reset button cannot be disabled, allowing an unstopable escape from any possible "hangup" condition.

These NMI interrupt functions are each separated in time (to prevent overlaps) and converted to pulses by the system hardware, in order to supply NMI transisitions required by the microprocessor logic.

*NOTE: That bit 5 is never disabled and therefore the Reset Button should not be pressed during power turn on.

IRQ Interrupts

IRQ interrupts are all "maskable" together by one bit of the status register on the microprocessor. This bit is set to the disable condition automatically by power turn on to prevent interrupt of power turn on service routines. In addition to this processor IRQ mask bit, there are separate system IRQ interrupt enable bits for each IRQ interrupt function (bits 0 thru 7 of IRQEN). These bits are not initalized by power turn on, and must be initalized by the program before enabling the processor IRQ. The 8 types of IRQ interrupts are;

D7 = BREAK KEY (depression of the break key)

D6 = OTHER KEY ( " of any other key)

D5 = SERIAL INPUT READY (Byte of serial data has been received and is ready to be read by the processor in SERIN register)

D4 = SERIAL OUTPUT NEEDED (Byte of serial data is being transmitted and SEROUT is ready to be written to again by the processor)

D3 = TRANSMISSION FINISHED (serial data transmission is finished. Output shift register is empty)

D2 = TIMER #4 (audio divider #4 has counted down to zero)

D1 = TIMER #2 (audio divider #2 has counted down to zero)

DO = TIMER #1 (audio divider #1 has counted down to zero)

In addition to the above IRQ interrupts (enabled by bits 0 thru 7 of IRQEN and identified by status bits 0 thru 7 of IRQST) there are two more system IRQ interrupts.

D7 of PACTL = peripheral "A" interrupt status bit

DO of PACTL = peripheral "A" interrupt enable bit

D7 of PBCTL = peripheral "B" interrupt status bit

DO of PBCTL = peripheral "B" interrupt enable bit

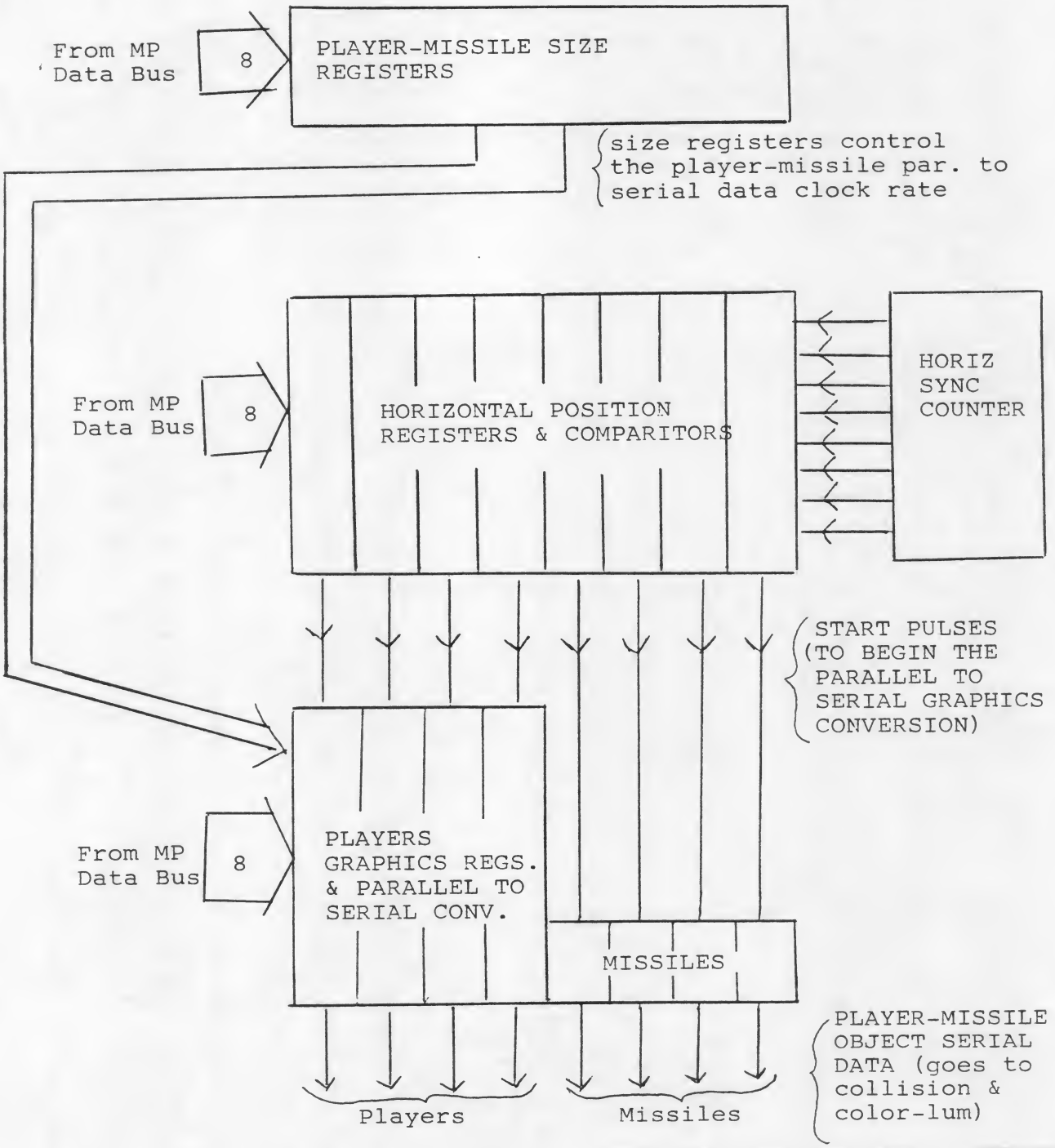
These last two interrupts are automatically disabled by power turn on, and their status bits are reset by reading from port A register and port B register. (See PORTA, PACTL, PORTB, PBCTL Register descriptions).

The IRQEN register, like the NMIEN register, enables interrupts when it's bits are 1 (logic true). The IRQST however (unlike the NMIST) has interrupt status bits that are normally logic true, and 90 to zero to indicate an interrupt request. The IRQST status bits are reset (returned to logic true) only by writing a zero into the corresponding IRQEN bit. This will disable the interrupt and simutanously reset (put true) the interrupt status bit. *Note, bit 3 of IRQST is not a latch and does not get reset by interrupt disable. It is zero when the serial out is empty (out finished) and true when it is not.

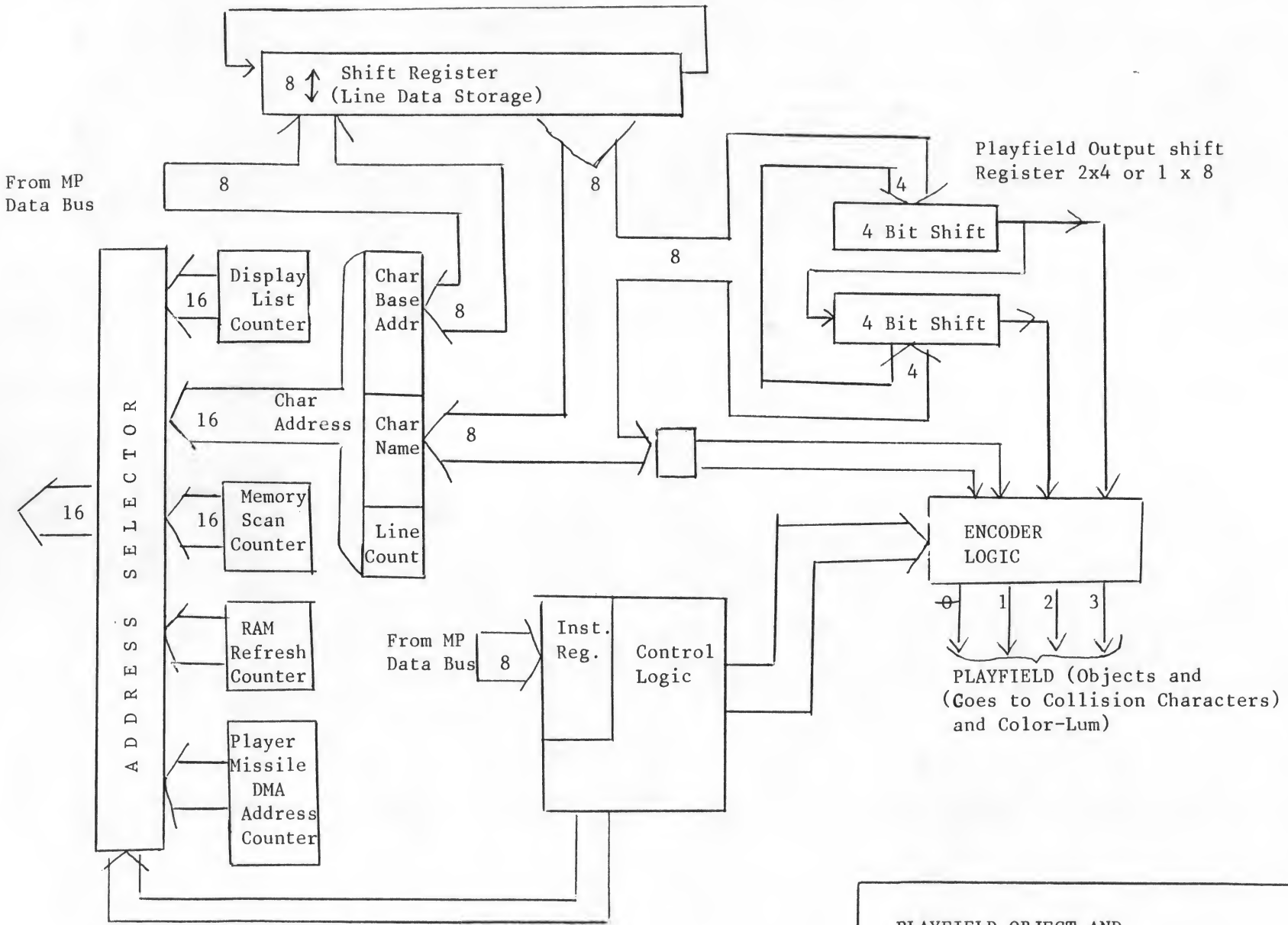
## I N T E R U P T  S U M M A R Y

| NAME | FUNCTIONS | ENABLE | STATUS | STATUS RESET |
|---|---|---|---|---|
| NMI INTERUPTS | Display<br>Instruct<br>Vert. Blk.<br>Reset Button | NMIEN<br>(Bits 6, thru 7)<br>Norm. Zero<br>(Disabled) | NMIST<br>(Bits 5thru7)<br>Norm. Zero<br>(no interrupt) | Address NMIRES<br>(Resets all NMI status together) |
| IRQ Interupts | KEYS<br>Serial ports<br>Timers | IRQEN<br>(Bits 0 thru7)<br>zero is<br>(Disabled)* | IRQST<br>(Bits 0thru7)<br>Norm. True<br>(no interrupt) | Reset (to true)<br>By Zero in Corresponding Bit of IRQEN<br>(except Bit 3) * |
| | Periph. A | D0 of PACTL<br>Norm. Zero<br>(Disabled) | D7 of PACTL<br>Norm Zero<br>(no interrupt) | Reset by Reading PortA Register |
| | Perip. B | D0 of PBLTL<br>Norm Zero<br>(Disabled) | D7 of PBCTL<br>Norm Zero<br>(no interrupt) | Reset by Reading PortB Register |

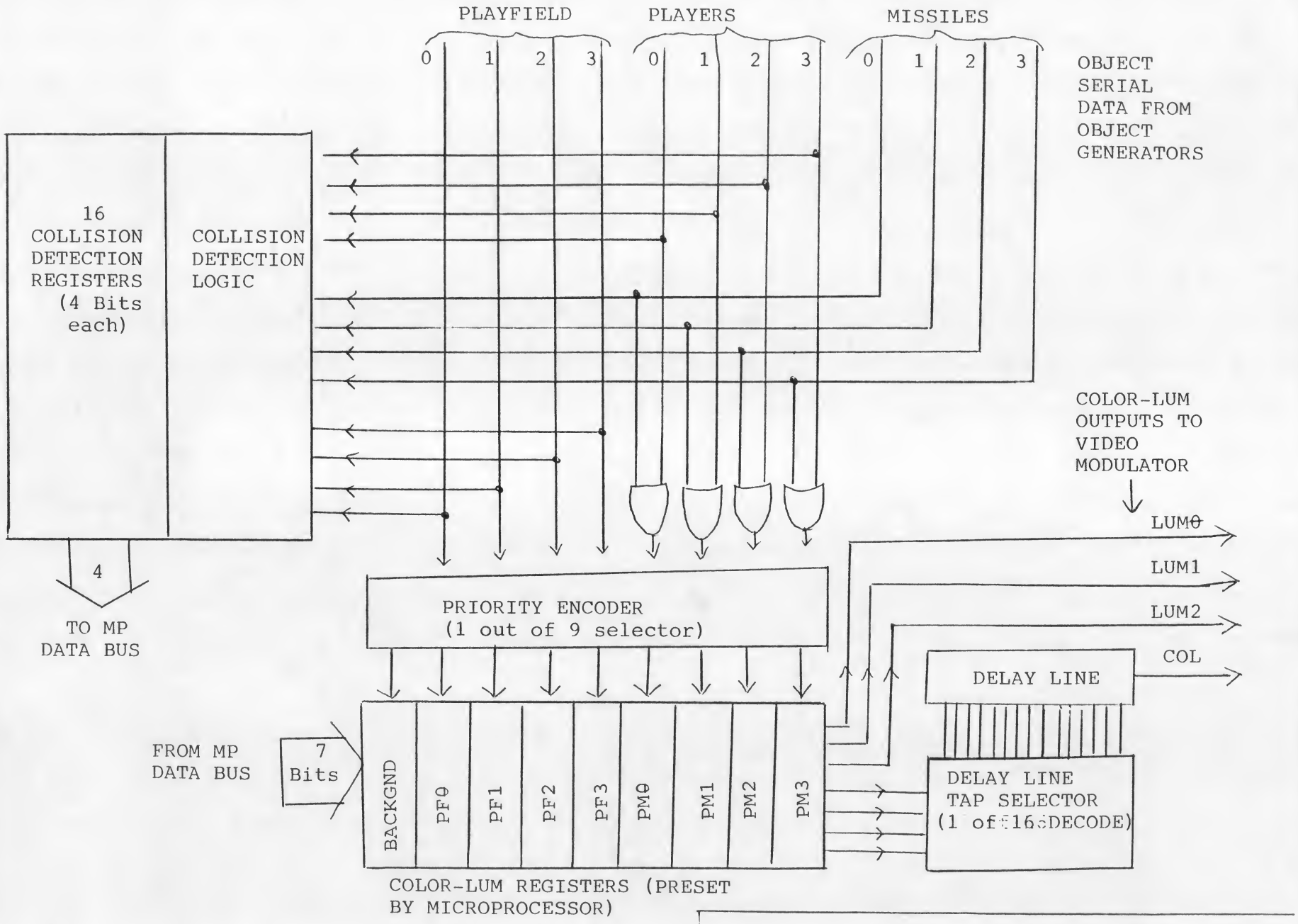
*Note: IRQEN is not automatically cleared at Power Turn On.



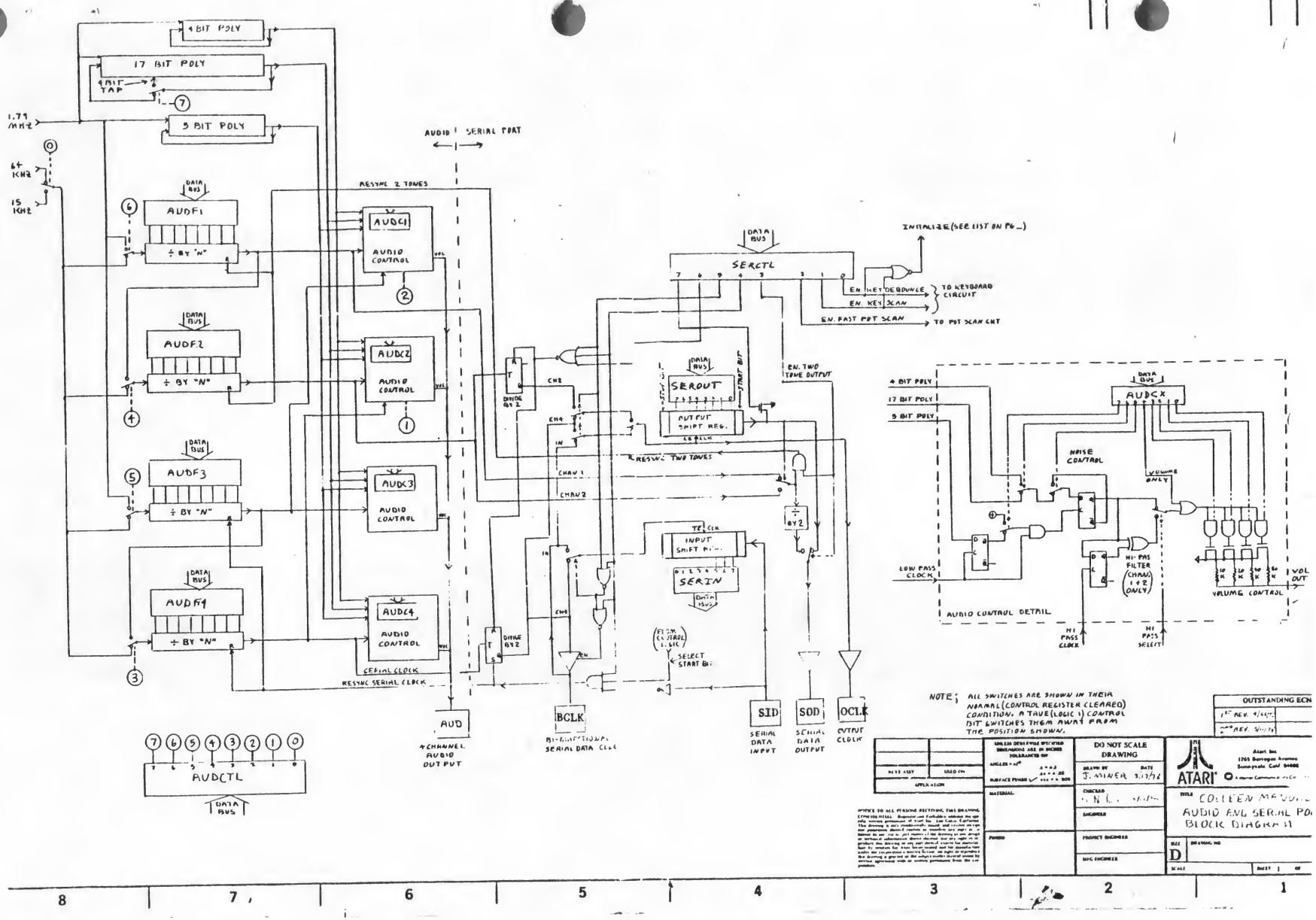

PLAYER-MISSILE
OBJECT GENERATORS
BLOCK DIAGRAM




PLAYFIELD OBJECT AND
CHARACTER GENERATOR
BLOCK DIAGRAM




COLLISION AND COLOR-LUM BLOCK DIAGRAM


4 BIT POLY
17 BIT POLY
4 BIT TAP
5 BIT POLY
1.79 MHZ
64 KHZ
15 KHZ
DATA BUS
AUDF1
AUDF2
AUDF3
AUDF4
÷ BY "N"
AUDC1
AUDC2
AUDC3
AUDC4
AUDIO CONTROL
AUDIO SERIAL PORT
RESYNC 2 TONES
SERCTL
INITIALIZE (SEE LIST ON PG_)
EN. KEY DEBOUNCE
EN. KEY SCAN
EN. FAST POT SCAN
TO KEYBOARD CIRCUIT
TO POT SCAN CKT
SEROUT
OUTPUT SHIFT REG.
START BIT
EN. TWO TONE OUTPUT
RESYNC TWO TONES
CHAN 1
CHAN 2
DIVIDE BY 2
÷ BY 2
INPUT SHIFT REG.
SERIN
SELECT START BIT
AUDCX
NOISE CONTROL
VOLUME ONLY
LOW PASS CLOCK
HI-PASS FILTER (CHAN 1 + 2 ONLY)
VOLUME CONTROL
AUDIO CONTROL DETAIL
HI PASS CLOCK
HI PASS SELECT
SERIAL CLOCK
RESYNC SERIAL CLOCK
AUD
4 CHANNEL AUDIO OUTPUT
BCLK
BI-DIRECTIONAL SERIAL DATA CLK
SID
SERIAL DATA INPUT
SOD
SERIAL DATA OUTPUT
OCLK
OUTPUT CLOCK
AUDCTL
NOTE: ALL SWITCHES ARE SHOWN IN THEIR NORMAL (CONTROL REGISTER CLEARED) CONDITION. A TRUE (LOGIC 1) CONTROL BIT SWITCHES THEM AWAY FROM THE POSITION SHOWN.
OUTSTANDING ECN
DO NOT SCALE DRAWING
DRAWN BY J. MINER
ATARI
Atari Inc.
1265 Borregas Avenue
Sunnyvale Calif 94086
TITLE COLLEEN MA
AUDIO AND SERIAL PO
BLOCK DIAGRAM
SIZE D
SHEET 1 OF
8
7
6
5
4
3
2
1

DMACTL (Direct Memory Access Control) D400

This address writes data into the DMA Control Register

DMA Control Register

| Not USED | | D5 | D4 | D3 | D2 | D1 | D0 | |
|---|---|---|---|---|---|---|---|---|
| X | X | 0 | 0 | 0 | 0 | 0 | 0 | DMA display inhibit * |
| | | | | | | 0 | 1 | Narrow (128) — Playfield DMA Enable |
| | | | | | | 1 | 0 | Standard (160) — Playfield DMA Enable |
| | | | | | | 1 | 1 | Wide (192) — Playfield DMA Enable |
| | | | | | 1 | | | Missile DMA Enable |
| | | | | 1 | | | | Player DMA Enable |
| | | | 1 | | | | | Player-Missile single line resolution |
| | | 1 | | | | | | Instruction fetch DMA enable |

* Note: All zero is power turn on condition

---

DLISTL (Display List Low) D402

This address writes data into the low Byte of the Display List Counter.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

← Display List Counter bit position

---

DLISTH (Display List High) D403

This address writes data into the high BYTE of the Display List Counter

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |

← Display List Counter bit position

The Display List is a list of display instructions in memory. These instructions are addressed by the Display List Counter. Loading these registers defines the address of the beginning of the Display List.

Note: The top 6 bits are latches only and have no count capability, therefore the Display List can not cross a 1K byte memory boundary without the use of a jump instruction.

CHACTL (Character Control) D401

This address writes data into the Character Control Register

| | | | |
|---|---|---|---|
| Not Used | D2 | D1 | D0 |

D2 Character Vertical Reflect Bit

This bit is sampled at the beginning of each line of characters. If true it causes the line of characters to reflect (invert) vertically.

D1 Character Video Invert Flag (used for 40 Char. Mode only) If bit 7 of character code is true this flag causes that character to be black on white.

D0 Character Blank (Blink) Flag (used for 40 char. mode only) If bit 7 of character code is true this flag causes that character to blank.

NMIEN (Non Maskable Interrupt Enable) D40E

This address writes data to the NMI interrupt enable bits. When these bits are zero the interrupts are disabled (masked).

| | | |
|---|---|---|
| D7 | D6 | Not Used |

D7 Instruction (Disp. List Inst.) Interrupt Enable
This bit is cleared by Power Reset, and may be set or cleared by the processor.

D6 Vert. Blank Inerrupt Enable
This bit is cleared by Power Reset, and may be set or cleared by the processor

Reset Button Interrupt

This interrupt is always enabled. Reset Button Should not be pressed during power turn on.

NMIST (Non Maskable Interrupt Status) D40F

This address reads the NMI Status Reg.

| D7 | D6 | D5 | Not Used |
|---|---|---|---|

D7 This bit identifies an NMI interrupt caused by bit 7 of a Display List Instruction

D6 This bit identifies an NMI interrupt caused by the beginning of vertical blank

D5 This bit identifies an NMI interrupt caused by the Reset Button.

---

NMIRES (NMI Status Reg. Reset) D40F

This write address resets the Non Maskable Interrupt Status Register (NMIST).

| Not Used |
|---|

---

CHBASE (Character Address Base Register) D409

This address writes data into the Character Address Base Register

D1 of Base Reg only used in 20 x 5 char mode

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | Not Used | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Base Reg. Char. Name Char. Line

CHARACTER ADDRESS

---

PMBASE (Player-Missile Address Base Register) D407

This address writes data into the Player-Missile Address Base Register

| D7 | D6 | D5 | D4 | D3 | D2 | Not Used | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

2 line resolution

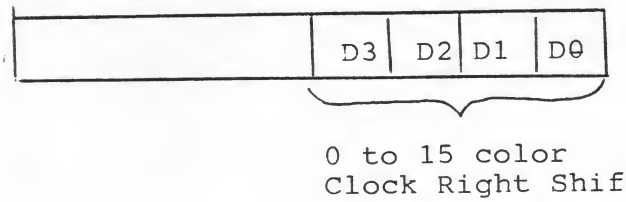
Base Reg. Player Missile Select Player-Missile Scan Counter

PLAYER-MISSILE ADDRESS

1 line resolution

HSCROL (Horizontal Scroll Register) D404

This address writes data into the Horizontal Scroll Register



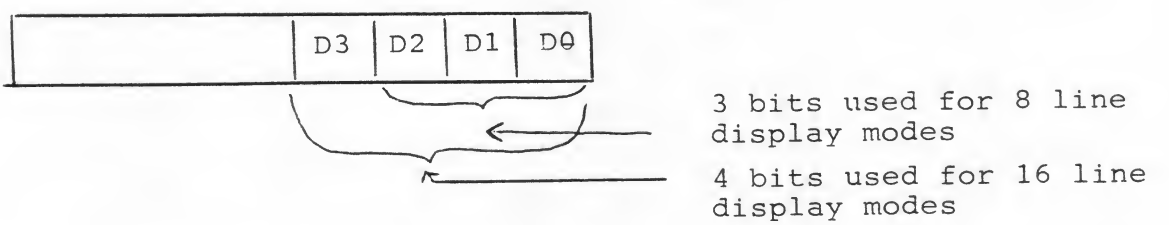

Note: This number defines the number of horiz. color clock right shifts of displayed data being horizontally scrolled. Data is horiz. scrolled if the Display List Instruction contains a1 in it's H SCROLL Flag bit (bit 4 of instruction Byte)

When horizontal scrolling is enabled, more bytes of data are needed. For a narrow playfield (see DMACTL bits 1 and 0) there should be the same number of bytes per line as for standard playfield with no scrolling. Similarly, for standard playfield use the same number of bytes as for the wide playfield, there is no change in the number of bytes and background color is shifted in. Players and missiles are not scrolled.

---

VSCROL (Vertical Scroll Register) D405

This address writes data into the Vertical Scroll Register




Note: This number defines the number of upward lines of vertical picture shift in any screen area being vertically scrolled. Data is vertically scrolled if the display list instruction contains a1 in its VSCROL Flag bit (bit 5 of instruction byte). The scrolled area will terminate with the first instruction having a zero in bit 5.

VCOUNT (Vertical Counter) *See note at bottom of page

This address reads the Vertical Counter
(8 most significant bits)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| V8 | V7 | V6 | V5 | V4 | V3 | V2 | V1 |

V0 not read. Two line resolution supplied.

D40B

---

D40D

PENV (Light Pen Vertical Value)

This address reads the Vertical Light Pen Register (8 most significant bits)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| LP8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

LP0 not read. two line resolution supplied

---

PENH (Light Pen Horiz. Value D40C

This address reads the Horizontal Light Pen Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| H7 | H6 | H5 | H4 | H3 | H2 | H1 | H0 |

---

WSYNC (Wait For Horz. Blank Synchronisim) D40A

| Not Used |
|---|

This address sets a latch that pulls down on the RDY line to the microprocessor, causing it to wait until this latch is automatically reset by the beginning of horizontal blank.

---

* NTSC

| VCOUNT | Line # | |
|---|---|---|
| 7C | 0 | vertical blank |
| 7D | 2 | |
| 7E | 4 | |
| 03 | 21 | |
| 04 | 22 | |
| 05 | 24 | |
| 7B | 261 | |

GRACTL (Graphics Control) D01D

This address writes data to the Graphic Control Register

| Not Used | D2 | D1 | D0 |
|---|---|---|---|

Note: DMACTL Register Also controls player-missile DMA

D2 Enable latches on TRIG 0 → TRIG 3 inputs (latches are cleared and TRIG 0 → TRIG 3 act as normal inputs when this control bit is zero)

D1 Enable player DMA to Player Graphics Regs.

D0 Enable Missile DMA to Missile Graphics Regs.

---

GRAFP0 → GRAFP3 (Player Graphics Registers)

These addresses write data directly into the Player Graphics Registers, independent of DMA.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

Left Right
Player on TV Screen

D00D
D00E
D00F
D010

---

GRAFM (Missile Graphics Register) D011

This address writes data directly into the Missile Graphics Register, Independent of DMA.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| L | R | L | R | L | R | L | R |
| M3 | | M2 | | M1 | | M0 | |

D00D
D00E
D00F
D010

---

HPOSP0 → HPOSP3 (Player Horiz. Position)

These addresses write data into the Player Horizontal Position Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | Left edge of screen |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | Right edge of screen |

D000
D001
D002
D003

HPOSM0 → HPOSM3 (Missile Horz. Position) D004 D005 D006 D007

These address writes data into the Missile Horizontal Position Registers

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | Left edge of screen |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | Right edge of screen |

---

SIZEP0 → SIZEP3 (Player Size) D008 D009 D00A D00B

These addresses write data into the Player Size Control Registers

| Not Used | D1 | D0 | Horz. Size Reg. (Player) |
|---|---|---|---|
| | 0 | 0 | Normal size |
| | 0 | 1 | Twice Size |
| | 1 | 0 | Normal size |
| | 1 | 1 | 4 times size |

---

D006

SIZEM (Missile Size)

This address writes data into the Missile Size Control Register.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
|---|---|---|---|---|---|---|---|---|
| M3 | | M2 | | M1 | | M0 | | Horz. Size Reg (Missile) |

| | | |
|---|---|---|
| 0 | 0 | Normal size |
| 0 | 1 | Twice size |
| 1 | 0 | Normal size |
| 1 | 1 | 4 times size |

---

VDELAY (Vertical Delay) D016

This address writes data into the Vertical Delay Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| P3 | P2 | P1 | P0 | M3 | M2 | M1 | M0 |

← When operating in the two line DMA mode these bits will move these objects down by one TV line

COLPM0 → COLPM3 (Player-Missile Color) D012 D013 D014 D015

These addresses write to the Player-Missile Color-Lum Registers. Missiles have the same color-lum as their player unless missiles are used as a 5th player (see bit 4 of PRIOR)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

(see "COLBAK" for bit assignments)

---

COLPF0 → COLPF3 (Playfield Color) D016 D017 D018 D019

These addresses write data to the Playfield Color-Lum Registers.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

(see COLBAK for bit assignment)

---

COLBAK (Background Color) D01A

This address writes data to the Background Color-Lum Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | Not Used | |
|---|---|---|---|---|---|---|---|---|
| X | X | X | X | 0 | 0 | 0 | | Zero luminance (black) |
| | | | | 0 | 0 | 0 | | |
| | | | | | ETC | | | |
| | | | | 1 | 1 | 1 | | Max. luminance (white) |
| 0 | 0 | 0 | 0 | | | | 1 | grey<br>gold |
| 0 | 0 | 1 | 0 | | | | 1 | orange<br>red-orange |
| 0 | 1 | 0 | 0 | | | | 1 | pink<br>purple |
| 0 | 1 | 1 | 0 | | | | 1 | purple-blue<br>blue |
| 1 | 0 | 0 | 0 | | | | 1 | blue<br>light blue |
| 1 | 0 | 1 | 0 | | | | 1 | turquoise<br>green-blue |
| 1 | 1 | 0 | 0 | | | | 1 | green<br>yellow-green |
| 1 | 1 | 1 | 0 | | | | 1 | orange-green<br>light orange |

PRIOR (Priority) D01B

This address writes data into the Priority Control Register

| | | | | | | |
|---|---|---|---|---|---|---|
| | D5 | D4 | D3 | D2 | D1 | D0 |

D5 Multi Color Player Enable.
This bit causes the logical "or" function of the bits of the colors of Player 0 with Player 1, and also of Player 2 with Player 3. This permits overlaping the position of 2 players with a choice of 3 colors in the overlaped region.

D4 Fifth Player Enable.
This bit causes all missiles to assume the color of Playfield Type 3. This allows missiles to be positioned together with a common color for use as a fifth player type object.

D3, D2, D1, D0 Priority Select (Mutually Exclusive)
These bits select one of 4 types of priority. Objects with higher priority will appear to move in front of objects with lower priority.

Higher Priority →

| D3=1 | D2=1 | D1=1 | D0=1 |
|---|---|---|---|
| PF0<br>PF1<br>P0<br>P1<br>P2<br>P3<br>PF2<br>PF3 + P5<br>BAK | PF0<br>PF1<br>PF2<br>PG3 + P5<br>P0<br>P1<br>P2<br>P3<br>BAK | P0<br>P1<br>PF0<br>PF1<br>PF2<br>PF3 + P5<br>P2<br>P3<br>BAK | P0<br>P1<br>P2<br>P3<br>PF0<br>PF!<br>PF2<br>PF3 + P5<br>BAK |

NOTE: The use of Priority bits in a "non exclusive" mode (more than 1 bit true) will result in objects (whose priorities are in conflict) turning BLACK in the overlap region.

EXAMPLE PRIOR code = 1010 This will black P0 or P1 if they are over PF0 or PF1. It will also black P2 or P3 if they are over PF2 or PF3.

TRIG0, TRIG1, TRIG2, TRIG3 (Trigger Ports) D010 D011 D012 D013

These addresses read port pins normally connected to the controller trigger buttons.

| | |
|---|---|
| Not Used (Zero Forced) | D0 |

(All are read in D0)
(Button zeros input)
*See Note Below

---

CONSOL (Console Switch Port) D01F

This address reads or writes data from the console switches and indicators

| | | | | |
|---|---|---|---|---|
| Not Used (zero Fored) | D3 | D2 | D1 | D0 |

Zeros must be written to this address in order to read the switches.

Ones written will pull down on the switch line.
* see CONSOL assignments below

---

*Note: TRIG 0 thru TRIG 3 are normally read directly by UP. However if bit 2 of GRACTL is 1 these inputs are latched whenever they go to logic zero. These latches are reset (true) when bit 2 of GRACTL=0.

---

*CONSOL Bit Assignment:

DØ ~~Game~~ Start
D1 ~~Game~~ Select
D2 Option ~~Select~~
D3 Loudspeaker

} Ø means switch pressed

M0PF, M1PF, M2PF, M3PF (Missile to Playfield Collisions) D000 D001 D002 D003

These addresses read Millile to Playfield Collisions.

| Not Used (zero forced) | D3 | D2 | D1 | D0 |
|---|---|---|---|---|
| | 3 | 2 | 1 | 0 |

Playfield Type

---

P0PF, P1PF, P2PF, P3PF (Player to Playfield Collisions) D004 D005 D006 D007

These addresses read Player to Playfield Collisions.

| Not Used (zero forced) | D3 | D2 | D1 | D0 |
|---|---|---|---|---|
| | 3 | 2 | 1 | 0 |

Playfield Type

---

M0PL, M1PL, M2PL, M3PL (Missile to Player Collision) D008 D009 D00A D00B

These addresses read Missile to Player Collisions

| Not Used (zero forced) | D3 | D2 | D1 | D0 |
|---|---|---|---|---|
| | 3 | 2 | 1 | 0 |

Player Type

---

P0PL, P1PL, P2PL, P3PL (Player to Player Collisions) D00C D00D D00E D00F

These addresses read Player to Player Collisions

| Not Used (zero forced) | D3 | D2 | D1 | D0 |
|---|---|---|---|---|
| | 3 | 2 | 1 | 0 |

Player Type

(Player 0 against Player 0 is always a zero). Etc.

---

HITCLR (Collision "HIT" Clear) D01E

This write address clears all collision bits described above

| Not Used |
|---|

POT0 → POT7 (Pot Values) D800-D807

These addresses read the value (1 to 228) of 8 pots connected to the 8 line pot port.

| D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|

Each Pot Value

They are valid only after 228 TV lines following the "POTGO" command described below.

---

ALL POT (All Pot Lines Simultaniously) D808

This address reads the present state of the 8 line pot port.

| D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|

8 Pot Line States

Capacitor dump transistors must be turned off by either going to fast pot scan mode (bit 2 of SERCTL) or starting pot scan (POTGO)

---

POTGO (Start Pot Scan) D80B

| No Data Bits Used |
|---|

This write address starts the pot scan sequence. The pot values (POT0 → POT7) should be read first. This write strobe is then used causing the following sequence.

1. Scan Counter cleared to zero.
2. Capacitor dump transistors turned off.
3. Scan Counter begins counting
4. Counter value captured in each of 8 registers (POT0 → POT7) as each pot line crosses trigger voltage.
5. Counter reaches 228, capacitor dump transistors turned on.

KBCODE (Keyboard Code) D209

This address reads the Keyboard Code, and is usually read in response to a Keyboard Interrupt (IRQ and bits 6 or 7 IRQST)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

D7 = Cntl Key

D6 = Shift Key

# KEYCODE TO ATASCII CONVERSION

| Key Code | Key Cap | L.C. | U.C. | CTRL | | Key Code | Key Cap | L.C. | U.C. | CTRL |
|---|---|---|---|---|---|---|---|---|---|---|
| 00 | L | 6C | 4C | 0C | | 20 | , | 2C | 5B | 00 |
| 01 | J | 6A | 4A | 0A | | 21 | SPACE | 20 | 20 | 20 |
| 02 | ; | 3B | 3A | 7B | | 22 | . | 2E | 5D | 60 |
| 03 | | | | | | 23 | N | 6E | 4E | 0E |
| 04 | | | | | | 24 | | | | |
| 05 | K | 6B | 4B | 0B | | 25 | M | 6D | 4D | 0D |
| 06 | + | 2B | 5C | 1E | | 26 | / | 2F | 3F | |
| 07 | * | 2A | 5E | 1F | | 27 | ^ | * | * | * |
| 08 | O | ~~2F~~ 6F | 4F | 0F | | 28 | R | 72 | 52 | 12 |
| 09 | | | | | | 29 | | | | |
| 0A | P | 70 | 50 | 10 | | 2A | E | 65 | 45 | 05 |
| 0B | U | 75 | 55 | 15 | | 2B | Y | 79 | 59 | 19 |
| 0C | RET | 9B | 9B | 9B | | 2C | TAB | 7F | 9F | 9E |
| 0D | I | 69 | 49 | 09 | | 2D | T | 74 | 54 | 14 |
| 0E | - | 2D | 5F | 1C | | 2E | W | 77 | 57 | 17 |
| 0F | = | 3D | 7C | 1D | | 2F | Q | 71 | 51 | 11 |
| 10 | V | 76 | 56 | 16 | | 30 | 9 | 39 | 28 | |
| 11 | | | | | | 31 | | | | |
| 12 | C | 63 | 43 | 03 | | 32 | 0 | 30 | 29 | |
| 13 | | | | | | 33 | 7 | 37 | 27 | |
| 14 | | | | | | 34 | BACKS | 7E | 9C | FE |
| 15 | B | 62 | 42 | 02 | | 35 | 8 | 38 | 40 | |
| 16 | X | 78 | 58 | 18 | | 36 | < | 3C | 7D | 7D |
| 17 | Z | 7A | 5A | 1A | | 37 | > | 3E | 9D | FF |
| 18 | 4 | 34 | 24 | | | 38 | F | 66 | 46 | 06 |
| 19 | | | | | | 39 | H | 68 | 48 | 08 |
| 1A | 3 | 33 | 23 | * | | 3A | D | 64 | 44 | 04 |
| 1B | 6 | 36 | 26 | | | 3B | | | | |
| 1C | ESC | 1B | 1B | 1B | | 3C | CAPS | * | * | * |
| 1D | 5 | 35 | 25 | | | 3D | G | 67 | 47 | 07 |
| 1E | 2 | 32 | 22 | FD | | 3E | S | 73 | 53 | 13 |
| 1F | 1 | 31 | 21 | * | | 3F | A | 61 | 41 | 01 |

* = special handling

SKCTL (Serial Port Control) D20F

This address writes data into the register that controls the configuration of the serial port, and also the Fast Pot Scan and Keyboard Enable.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

(Bits are normally zero and perform the functions shown below when true)

D7 Force Break (force serial output to zero (space)*

D6, D5, D4 Serial Port Mode Control (see mode chart at end of Serial port description.) p.19

D3 Two Tone (Serial output transmitted as two tone signal instead of logic true/false).

D2 Fast Pot (Fast Pot Scan. The Pot Scan Counter completes it's sequence in two TV line times instead of one frame time. The capacitor dump transistors are completely disabled.)

D1 Enable Key Scan (Enables Keyboard Scanning circuit)

D0 Enable Debounce (Enables Keyboard Debounce circuits)

$\overline{D0}$-$\overline{D1}$ (Both zero) Initalize (State used for testing and initalizing chip) **

*NOTE: When powered on, serial port output may stay low even if this bit is cleared. To get S.P. high (mark), send a byte out (recommend ØØ or FF).

**NOTE: There is no original power on state. Pokey has no reset pin.

D20D

SERIN (Serial Input Data)

This address reads the 8 bit parallel holding register is loaded when a full byte of serial input data has been received. This address is usually read in response to a serial data in interrupt (IRQ and bit 5 of IRAST)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

---

SEROUT (Serial Output Data) D20D

This address writes to the 8 bit parallel holding register that is transferred to the output serial shift register when a full byte of serial output data has been transmitted. This address is usually written in response to a serial data out interrupt (IRQ and bit 4 of IRQST)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

---

IRQST (IRQ Interrupt Status)

This address reads the data from the IRQ Interrupt Status Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

These bits are normally 1 (true) and go to zero to indicate they following Functions.

D7 = 0 Break Key Interrupt
D6 = 0 Other Key Interrupt
D5 = 0 Serial Input Data Ready Interrupt
D4 = 0 Serial Output Data Needed Interrupt
D3 = 0 Serial Output (Byte) Transmission Finished Interrupt *
D2 = 0 Timer 4 Interrupt
D1 = 0 Timer 2 Interrupt
D0 = 0 Timer 1 Interrupt

* Note: Used for generating of 2 stop bits. See IRQ description on Pg. 22 (no direct reset on bit 3).

---

IRQEN (IRQ Interrupt Enable) D20E

This address writes data to the IRQ Interrupt Enable bits. When these bits are zero the interrupts are disabled (masked) and and the corresponding bit of the IRQST (IRQ Interrupt Status Reg) is reset (to logic true)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

D7 Break Key Interrupt Enable
D6 Other Key Interrupt Enable
D5 Serial Input Data Ready Interrupt Enable
D4 Serial Output Data Needed Interrupt Enable
D3 Serial Out Trans. Finished Interrupt Enable
D2 Timer 4 Interrupt Enable
D1 Timer 2 Interrupt Enable
D0 Timer 1 Interrupt Enable

SKSTAT (Serial Port-Keyboard Status) D20F

This address reads the status register giving information about the serial port and keyboard.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

(Bits are normally true and provide the following information when zero)

D7 = 0 = Serial Data Input Frame Error
D5 = 0 = Serial Data Input Over-run
D6 = 0 = Keyboard Over-run
D4 = 0 = Direct from Serial Input Port
D3 = 0 = Shift Key Depressed
D2 = 0 = Last Key is Still Depressed
D1 = 0 = Serial Input Shift Reg. Busy
D0 = Not Used (Logic True)

Latches must be reset = 1 (SKRES)

(D5 and D6 are set to zero when new data and same bit of IRQST is zero)

---

SKRES (Reset above Status Register D20A

This write address resets bits 7, 6, and 5 of the Serial Port-Keyboard Status Register to 1.

| No Data Bits |
|---|

---

RANDOM (Random Number Generator) D20A

This address reads the high order 8 bits of a 17 bit polynomial counter (9 bit, if bit 7 of AUDCTL=1)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

---

STIMER (Start Timer) D209

This write address resets all audio frequency dividers to their "AUDF" value. These dividers generate timer interrupts when they count down to zero (If enabled IRQEN).

| No Data Bits |
|---|

AUDCTL (Audio Control) D208

This address writes data into the Audio Mode Control Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

These data register bits control audio functions described below

BIT 7 Change 17 bit poly into a 9 bit poly

BIT 6 Clock Chan 1 with 1.79 MHZ, Instead of 64 KHZ

BIT 5 Clock Chan 3 with 1.79 MHZ, instead of 64 KHZ

BIT 4 Clock Chan 2 with Chan 1, instead of 64 KHZ (16 BIT)

BIT 3 Clock Chan 4 with Chan 3, instead of 64 KHZ (16 BIT)

BIT 2 Insert Hi Pass filter in Chan 1, clocked by Chan 3. (see p. 14)

BIT 1 Insert Hi Pass filter in Chan 2, clocked by Chan 4.

Bit 0 Change Normal 64 KHZ freq, into 15 KHZ.

EXACT FREQUENCIES

The frequencies given above are approximate. The Exact Frequency (fin) that clocks the divide by N counters is given below (NTSC only, PAL different)

| Fin (Approx) | Fin (Exact) | |
|---|---|---|
| 1.79 MHZ | 1.78979 MHZ | - Use modified formula for fout |
| 64 KHZ | 63.9210 KHZ | Use normal formula for fout |
| 15 KHZ | 15.6999 KHZ | |

The Normal Formula for output frequency is,

$$\boxed{Fout = Fin/2n}$$

Where N = The binary number in the freq. Reg (AUDF), plus 1. (N=AUDF+1)

The MODIFIED FORMULA should be used when Fin = 1.79 MHZ and a more exact result is desired,

$$\boxed{Fout = \frac{Fin}{2(AUDF+M)}}$$

Where: M = 4 if 8 bit counter (AUDCTL bit 3 or 4 = 0)
M = 7 if 16 bit counter (AUDCTL bit 3 or 4 = 1)

AUDF1, AUDF2, AUDF 3, AUDF 4 (Audio Frequency) C8

These addresses write data into each of the four Audio Frequency Control Registers. Each register controls a divide by "N" counter

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | "N" |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| | | | ETC | | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 255 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 256 |

Note: "N" is one greater than the binary number in Audio Frequency Register AUDF(X)

D200
D202
D204
D206

---

AUDC1, AUDC2, AUDC3, AUDC4 (Audio Channel Control)

These addresses write data into each of the four Audio Control Registers. Each Register controls the noise content and volume of the corresponding Audio Channel

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Divisor "N" set by audio Frequency register |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | 17 BIT poly ÷ 5 BIT poly ÷ N |
| 0 | 0 | 1 | 0 | | | | | 5 BIT poly ÷ N ÷ 2 |
| 0 | 1 | 0 | 0 | | | | | 4 BIT poly ÷ 5 bit poly ÷ N |
| 0 | 1 | 1 | 0 | | | | | 5 BIT poly ÷ N ÷ 2 |
| 1 | 0 | 0 | 0 | | | | | 17 BIT poly ÷ N |
| 1 | X | 1 | 0 | | | | | Pure Tone ÷ N ÷ 2 |
| 1 | 1 | 0 | 0 | | | | | 4 BIT poly ÷ N |
| X | X | X | 1 | | | | | Force Output (volume only) |
| | | | X | 0 | 0 | 0 | 0 | Lowest Volume (Off) |
| | | | X | 1 | 0 | 0 | 0 | Half Volume |
| | | | X | 1 | 1 | 1 | 1 | Highest volume |

D201
D203
D205
D207

PORT A (Port A) D300

This address reads or writes data from Player 0 and Player 1 controller jacks if Bit 2 of PACTL is true. This address writes to the direction control register if bit 2 of PACTL is zero

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| R | L | B | F | R | L | B | F |
| Jack1 | | | | JACK0 | | | |

PortA Register
(Addressed if bit 2 of PACTL is true)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

Direction of Jack Pins
0=Input 1=Output)

Direction A Control Register
(Addressed if bit 2 of PACTL is zero)

---

PACTL (Port A Control) D302

This address writes or reads data from the Port A Control Register

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| X | 0 | 1 | 1 | X | X | 0 | X |

Port A Control Reg.

Set up register as shown
(x=described below)

D2 Controls PortA addressing described above
(1 = Port A Register 0=Direction Control Reg.)

D3 Peripheral Motor Control
(0=On 1=Off)

D7 (Read only) Perph. A Interrupt Status Bit
(Reset by reading Port A Register)
Set by Peripheral A Interrupt)

D0 Peripheral A Interrupt Enable Bit
(Reset by power turn on or processor)
(Set by Processor)

PORTB (Port B)

This address reads or writes data from Player 2 and Player 3 controller jacks if bit 2 of PBCTL is true. This address writes to the Direction Control Register if bit 2 of PBCTL is zero

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| R | L | B | F | R | L | B | F |
| Jack3 | | | | Jack2 | | | |

Port B Register
(Addressed if bit 2)
of PBCTL is true.)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

Direction of Jack pins
(0=input 1=output)

Direction B Control
Register
(Addressed if bit 2 of
PBCTL is zero)

---

PBCTL (Port B Control) D303

This address writes or reads data from the Port B Control Register.

Read only

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| X | 0 | 1 | 1 | X | X | 0 | X |

Port B Control Reg

Set up register as shown
(X=Described below)

D2 Controls Port B addressing described above
(1=Port B Register 0=Direction Control Reg.)

D3 Peripheral Command Identification (COMMAND)

D7 (Read only) Peripheral B Interrupt Status Bit
(Reset by Reading Port B Register)
(Set by Peripheral B Interrupt)

D0 Peripheral B Interrupt Enable Bit
(Reset by power turn on or processor)
(set by processor)

# ATARI 400 and 800

## MEMORY MAP

| ADDRESS | FUNCTION | SIZE |
|---|---|---|
| FFFF<br>D800 | Operating System And Math Routines | 10K |
| D000-D7FF | Hardware Addresses | 2K |
| CFFF<br>C000 | Reserved for Future O.S. expansion | 4K |
| BFFF<br>8000 | ROM Cartridge<br>(Colleen left and right slot and Candy single slot all address to this space) | 16K |
| 7FFF<br>2000 | RAM<br>Expansion * | |
| 1FFF<br>0000 | RAM initially supplied in the product | 8K |

* RAM expansion can actually extend to BFFF. However, the ROM cartridges will deselect the RAM. Deselection occurs on 8K boundaries. Atari 400 units are RAM expandable only at the factory. They can accept RAM up to 2FFF (16K) when fully extended. It is presently planned, though to only sell 400's with 8K.

## CTIA ADDRESSES

| Address | WRITE Name | WRITE Description | READ NAME | READ Description |
|---|---|---|---|---|
| D1FF ↕ D020 | REPEAT AS BELOW | | 15 MORE Times | |
| D01F | CONSOL | Write Consol SW.PORT | CONSOL | Read Consol SW.PORT |
| D01E | HITCLR | Collision Clear | | |
| D01D | GRACTL | Graphics Control | | |
| D01C | VDELAY | Vert. Delay | | |
| D91B | PRIOR | Priority Select | | |
| D01A | COLBK | Col-lum Bkgnd | | |
| D019 | COLPF3 | Color-lum of 3 Playfield | | |
| D018 | COLPF2 | 2 | | |
| D017 | COLPF1 | Playfield 1 | | |
| D016 | COLPF0 | Playfield 0 | | |
| D015 | COLPM3 | Color-lum of 3 | | |
| D014 | COLPM2 | Player-Missile 2 | | |
| D013 | COLPM1 | Player-Missile 1 | TRIG3 | Read Controller Trigger Buttons |
| D012 | COLPM0 | Player-Missile 0 | TRIG2 | |
| D011 | GRAFM | Graphics All Missiles | TRIG1 | |
| D010 | GRAFP3 | Graphics Player 3 | TRIG0 | |
| D00F | GRAFP2 | Graphics Player 2 | P3PL | Read Player To Player Collisions |
| D00E | GRAFP1 | Graphics Player 1 | P2PL | |
| D00D | GRAFP0 | Graphics Player 0 | P1PL | |
| D00C | SIZEM | Size All Missiles | P0PL | |
| D00B | SIZEP3 | Size Player 3 | M3PL | Read Missile To Player Collisions |
| D00A | SIZEP2 | Size Player 2 | M2PL | |
| D009 | SIZEP1 | Size Player 1 | M1PL | |
| D008 | SIZEP0 | Size Player 0 | M0PL | |
| D007 | HPOSM3 | Horz. Posit. Missile3 | P3PF | Read Player To Playfield Collisions |
| D006 | HPOSM2 | Horz. Posit. Missile2 | P2PF | |
| D005 | HPOSM1 | Horz. Posit. Missile1 | P1PF | |
| D004 | HPOSM0 | Horz. Posit. Missile0 | P0PF | |
| D003 | HPOSP3 | Horz. Posit. Player 3 | M3PF | Read Missile To Playfield Collisions |
| D002 | HPOSP2 | Horz. Posit. Player 2 | M2PF | |
| D001 | HPOSP1 | Horz. Posit. Player 1 | M1PF | |
| D000 | HPOSP0 | Horz. Posit. Player 0 | M0PF | |

ANTIC ADDRESSES

| Address | WRITE Name | WRITE Description | READ Name | READ Description |
|---|---|---|---|---|
| D4FF ↕ D410 | Repeat (As Below) | | 15 More Times | |
| D40F | NMIRES | Reset NMI Interrupt Status | NMIST | NMI Interrupt Status Req. |
| D40E | NMIEN | NMI Interrupt ENABLE | | |
| D40D | | | PENV | Light Pen Reg. Vert. |
| D40C | | | PENH | Light Pen Reg. Horz. |
| D40B | | | VCOUNT | Vertical Line Counter |
| D40A | WSYNC | Wait for HBLANK Synchronisim | | |
| D409 | CHBASE | Character Base Address Req | | |
| D408 | | | | |
| D407 | PMBASE | Player-Missile Base Address Req. | | |
| D406 | | | | |
| D405 | VSCROLL | Vert. Scroll Req. | | |
| D404 | HSCROLL | Horiz. Scroll Req. | | |
| D403 | DLISTH | Display List Pointer (High Byte) | | |
| D402 | DLISTL | Display List Pointer (Low Byte) | | |
| D401 | CHACTL | Character Control Req. | | |
| D400 | DMACTL | DMA Control Req. | | |
| | | | | |
| | | | | |
| | | | | |

POKEY ADDRESSES

| | WRITE | | READ | |
|---|---|---|---|---|
| | Name | Description | Name | Description |
| D2FF ↕ D210 | Repeat As below | | 15 More Times | |
| D20F | SKCTL | Serial Port 4 Key Control | SKSTAT | Serial Port 4 Key Status Req. |
| D20E | IRQEN | IRQ Interrupt Enable | IRQST | IRQ Interrupt Status Req |
| D20D | SEROUT | Serial Port Output Req. | SERIN | Serial Port Input Req. |
| D20C | | | | |
| D20B | POTGO | Start Pot Scan Sequence | | |
| D20A | SKRES | Reset Status (SKSTAT) | RANDOM | Random Numb. Generator |
| D209 | STIMER | Start Timers | KBCODE | Keyboard Code |
| D208 | AUDCTL | Audio Control | ALLPOT | Read 8 Line Pot Port State |
| D207 | AUDC4 | Audio Chan. 4 Control | POT 7 | Read the value of each POT |
| D206 | AUDF4 | Audio Chan. 4 Frequency | POT6 | |
| D205 | AUDC3 | Audio Chan. 4 Control | POT5 | |
| D204 | AUDF3 | Audio Chan. 3 Frequency | POT4 | |
| D203 | AUDC2 | Audio Chan. 2 Control | POT 3 | |
| D202 | AUDF2 | Audio Chan. 2 Frequency | POT 2 | |
| D201 | AUDC1 | Audio Chan. 1 Control | POT 1 | |
| D200 | AUDF1 | Audio Chan 1 Frequency | POT 0 | |
| | | | | |
| | | | | |

PIA ADDRESSES

| Address | WRITE | | READ | |
|---|---|---|---|---|
| | Name | Description | Name | Description |
| D3FF<br>D304 | Repeat as shown below many times | | | |
| D303 | PBCTL | Port B Control | PBCTL | Same as write |
| D302 | PACTL | Port A Control | PACTL | Same as write |
| D301 | PORTB | Direction Register If PBCTL Bit 2-0 (otherwise ) | PORTB | Same as write |
| | PORTB | Jack 2 & Jack 3 If Direction Bits Are 1 * | PORTB | Jack 2 & Jack 3 If Direction Bits Are 0 * |
| D300 | PORTA | Direction Register If PACTL Bit 2=0 (Otherwise ) | PORTA | Same as write |
| | PORTA | Jack 0 & Jack 1 If Direction Bits Are 1 * | PORTA | Jack 0 & Jack 1 If Direction Bits Are 0 * |
| | | | | |
| | | | | |

* NOTE: Output data is retained in Jack Output Registers. If direction bits are true, a read of the jacks will read old data from these registers.